Objects

In this article

About objects
ActorLocation
AddedToMergeQueueEvent
AddedToProjectEvent
App
AssignedEvent
AutoMergeDisabledEvent
AutoMergeEnabledEvent
AutoMergeRequest
AutoRebaseEnabledEvent
AutoSquashEnabledEvent
AutomaticBaseChangeFailedEvent
AutomaticBaseChangeSucceededEvent
BaseRefChangedEvent
BaseRefDeletedEvent
BaseRefForcePushedEvent
Blame
BlameRange
Blob
Bot
BranchNamePatternParameters
BranchProtectionRule
BranchProtectionRuleConflict
BranchProtectionRuleConflictConnection
BranchProtectionRuleConflictEdge
BranchProtectionRuleConnection
BranchProtectionRuleEdge
BypassForcePushAllowance
BypassForcePushAllowanceConnection
BypassForcePushAllowanceEdge
BypassPullRequestAllowance
BypassPullRequestAllowanceConnection
BypassPullRequestAllowanceEdge
CVSS
CWE
CWEConnection
CWEEdge
CheckAnnotation
CheckAnnotationConnection
CheckAnnotationEdge
CheckAnnotationPosition
CheckAnnotationSpan
CheckRun

CheckRunConnection
CheckRunEdge
CheckRunStateCount
CheckStep
CheckStepConnection
CheckStepEdge
CheckSuite
CheckSuiteConnection
CheckSuiteEdge
ClosedEvent
CodeOfConduct
CommentDeletedEvent
Commit
CommitAuthorEmailPatternParameters
CommitComment
CommitCommentConnection
CommitCommentEdge
CommitCommentThread
CommitConnection
CommitContributionsByRepository
CommitEdge
CommitHistoryConnection
CommitMessagePatternParameters
CommitterEmailPatternParameters
Comparison
ComparisonCommitConnection
ConnectedEvent
ContributingGuidelines
ContributionCalendar
ContributionCalendarDay
ContributionCalendarMonth
ContributionCalendarWeek
ContributionsCollection
ConvertToDraftEvent
ConvertedNoteToIssueEvent
ConvertedToDiscussionEvent
CreatedCommitContribution
CreatedCommitContributionConnection
CreatedCommitContributionEdge
CreatedIssueContribution
CreatedIssueContributionConnection
CreatedIssueContributionEdge
CreatedPullRequestContribution
CreatedPullRequestContributionConnection
CreatedPullRequestContributionEdge
CreatedPullRequestReviewContribution
CreatedPullRequestReviewContributionConnection
CreatedPullRequestReviewContributionEdge
CreatedRepositoryContribution
CreatedRepositoryContributionConnection
CreatedRepositoryContributionEdge

CrossReferencedEvent
DemilestonedEvent
DependabotUpdate
DependabotUpdateError
DependencyGraphDependency
DependencyGraphDependencyConnection
DependencyGraphDependencyEdge
DependencyGraphManifest
DependencyGraphManifestConnection
DependencyGraphManifestEdge
DeployKey
DeployKeyConnection
DeployKeyEdge
DeployedEvent
Deployment
DeploymentConnection
DeploymentEdge
DeploymentEnvironmentChangedEvent
DeploymentProtectionRule
DeploymentProtectionRuleConnection
DeploymentProtectionRuleEdge
DeploymentRequest
DeploymentRequestConnection
DeploymentRequestEdge
DeploymentReview
DeploymentReviewConnection
DeploymentReviewEdge
DeploymentReviewerConnection
DeploymentReviewerEdge
DeploymentStatus
DeploymentStatusConnection
DeploymentStatusEdge
DisconnectedEvent
Discussion
DiscussionCategory
DiscussionCategoryConnection
DiscussionCategoryEdge
DiscussionComment
DiscussionCommentConnection
DiscussionCommentEdge
DiscussionConnection
DiscussionEdge
DiscussionPoll
DiscussionPollOption
DiscussionPollOptionConnection
DiscussionPollOptionEdge
DraftIssue
Enterprise
EnterpriseAdministratorConnection
EnterpriseAdministratorEdge
EnterpriseAdministratorInvitation

EnterpriseAdministratorInvitationConnection
EnterpriseAdministratorInvitationEdge
EnterpriseBillingInfo
EnterpriseConnection
EnterpriseEdge
EnterpriseFailedInvitationConnection
EnterpriseFailedInvitationEdge
EnterpriseIdentityProvider
EnterpriseMemberConnection
EnterpriseMemberEdge
EnterpriseOrganizationMembershipConnection
EnterpriseOrganizationMembershipEdge
EnterpriseOutsideCollaboratorConnection
EnterpriseOutsideCollaboratorEdge
EnterpriseOwnerInfo
EnterprisePendingMemberInvitationConnection
EnterprisePendingMemberInvitationEdge
EnterpriseRepositoryInfo
EnterpriseRepositoryInfoConnection
EnterpriseRepositoryInfoEdge
EnterpriseServerInstallation
EnterpriseServerInstallationConnection
EnterpriseServerInstallationEdge
EnterpriseServerInstallationMembershipConnection
EnterpriseServerInstallationMembershipEdge
EnterpriseServerUserAccount
EnterpriseServerUserAccountConnection
EnterpriseServerUserAccountEdge
EnterpriseServerUserAccountEmail
EnterpriseServerUserAccountEmailConnection
EnterpriseServerUserAccountEmailEdge
EnterpriseServerUserAccountsUpload
EnterpriseServerUserAccountsUploadConnection
EnterpriseServerUserAccountsUploadEdge
EnterpriseUserAccount
Environment
EnvironmentConnection
EnvironmentEdge
ExternalIdentity
ExternalIdentityAttribute
ExternalIdentityConnection
ExternalIdentityEdge
ExternalIdentitySamlAttributes
ExternalIdentityScimAttributes
FollowerConnection
FollowingConnection
FundingLink
GenericHovercardContext
Gist
GistComment
GistCommentConnection

GistCommentEdge
GistConnection
GistEdge
GistFile
GitActor
GitActorConnection
GitActorEdge
GitHubMetadata
GpgSignature
HeadRefDeletedEvent
HeadRefForcePushedEvent
HeadRefRestoredEvent
Hovercard
IpAllowListEntry
IpAllowListEntryConnection
IpAllowListEntryEdge
Issue
IssueComment
IssueCommentConnection
IssueCommentEdge
IssueConnection
IssueContributionsByRepository
IssueEdge
IssueTemplate
IssueTimelineConnection
IssueTimelineItemEdge
IssueTimelineItemsConnection
IssueTimelineItemsEdge
JoinedGitHubContribution
Label
LabelConnection
LabelEdge
LabeledEvent
Language
LanguageConnection
LanguageEdge
License
LicenseRule
LinkedBranch
LinkedBranchConnection
LinkedBranchEdge
LockedEvent
Mannequin
MannequinConnection
MannequinEdge
MarkedAsDuplicateEvent
MarketplaceCategory
MarketplaceListing
MarketplaceListingConnection
MarketplaceListingEdge
MembersCanDeleteReposClearAuditEntry

MembersCanDeleteReposDisableAuditEntry

MembersCanDeleteReposEnableAuditEntry

MentionedEvent

MergeQueue

MergeQueueConfiguration

MergeQueueEntry

MergeQueueEntryConnection

MergeQueueEntryEdge

MergedEvent

MigrationSource

Milestone

MilestoneConnection

MilestoneEdge

MilestonedEvent

MovedColumnsInProjectEvent

OIDCProvider

OauthApplicationCreateAuditEntry

OrgAddBillingManagerAuditEntry

OrgAddMemberAuditEntry

OrgBlockUserAuditEntry

OrgConfigDisableCollaboratorsOnlyAuditEntry

OrgConfigEnableCollaboratorsOnlyAuditEntry

OrgCreateAuditEntry

OrgDisableOauthAppRestrictionsAuditEntry

OrgDisableSamlAuditEntry

OrgDisableTwoFactorRequirementAuditEntry

OrgEnableOauthAppRestrictionsAuditEntry

OrgEnableSamlAuditEntry

OrgEnableTwoFactorRequirementAuditEntry

OrgInviteMemberAuditEntry

OrgInviteToBusinessAuditEntry

OrgOauthAppAccessApprovedAuditEntry

OrgOauthAppAccessBlockedAuditEntry

OrgOauthAppAccessDeniedAuditEntry

OrgOauthAppAccessRequestedAuditEntry

OrgOauthAppAccessUnblockedAuditEntry

OrgRemoveBillingManagerAuditEntry

OrgRemoveMemberAuditEntry

OrgRemoveOutsideCollaboratorAuditEntry

OrgRestoreMemberAuditEntry

OrgRestoreMemberMembershipOrganizationAuditEntryData

OrgRestoreMemberMembershipRepositoryAuditEntryData

OrgRestoreMemberMembershipTeamAuditEntryData

OrgUnblockUserAuditEntry

OrgUpdateDefaultRepositoryPermissionAuditEntry

OrgUpdateMemberAuditEntry

OrgUpdateMemberRepositoryCreationPermissionAuditEntry

OrgUpdateMemberRepositoryInvitationPermissionAuditEntry

Organization

OrganizationAuditEntryConnection

OrganizationAuditEntryEdge

OrganizationConnection
OrganizationEdge
OrganizationEnterpriseOwnerConnection
OrganizationEnterpriseOwnerEdge
OrganizationIdentityProvider
OrganizationInvitation
OrganizationInvitationConnection
OrganizationInvitationEdge
OrganizationMemberConnection
OrganizationMemberEdge
OrganizationMigration
OrganizationTeamsHovercardContext
OrganizationsHovercardContext
Package
PackageConnection
PackageEdge
PackageFile
PackageFileConnection
PackageFileEdge
PackageStatistics
PackageTag
PackageVersion
PackageVersionConnection
PackageVersionEdge
PackageVersionStatistics
PageInfo
PermissionSource
PinnableItemConnection
PinnableItemEdge
PinnedDiscussion
PinnedDiscussionConnection
PinnedDiscussionEdge
PinnedEvent
PinnedIssue
PinnedIssueConnection
PinnedIssueEdge
PrivateRepositoryForkingDisableAuditEntry
PrivateRepositoryForkingEnableAuditEntry
ProfileItemShowcase
Project
ProjectCard
ProjectCardConnection
ProjectCardEdge
ProjectColumn
ProjectColumnConnection
ProjectColumnEdge
ProjectConnection
ProjectEdge
ProjectProgress
ProjectV2
ProjectV2ActorConnection

ProjectV2ActorEdge
ProjectV2Connection
ProjectV2Edge
ProjectV2Field
ProjectV2FieldConfigurationConnection
ProjectV2FieldConfigurationEdge
ProjectV2FieldConnection
ProjectV2FieldEdge
ProjectV2Item
ProjectV2ItemConnection
ProjectV2ItemEdge
ProjectV2ItemFieldDateValue
ProjectV2ItemFieldIterationValue
ProjectV2ItemFieldLabelValue
ProjectV2ItemFieldMilestoneValue
ProjectV2ItemFieldNumberValue
ProjectV2ItemFieldPullRequestValue
ProjectV2ItemFieldRepositoryValue
ProjectV2ItemFieldReviewerValue
ProjectV2ItemFieldSingleSelectValue
ProjectV2ItemFieldTextValue
ProjectV2ItemFieldUserValue
ProjectV2ItemFieldValueConnection
ProjectV2ItemFieldValueEdge
ProjectV2IterationField
ProjectV2IterationFieldConfiguration
ProjectV2IterationFieldIteration
ProjectV2SingleSelectField
ProjectV2SingleSelectFieldOption
ProjectV2SortBy
ProjectV2SortByConnection
ProjectV2SortByEdge
ProjectV2SortByField
ProjectV2SortByFieldConnection
ProjectV2SortByFieldEdge
ProjectV2View
ProjectV2ViewConnection
ProjectV2ViewEdge
ProjectV2Workflow
ProjectV2WorkflowConnection
ProjectV2WorkflowEdge
PublicKey
PublicKeyConnection
PublicKeyEdge
PullRequest
PullRequestChangedFile
PullRequestChangedFileConnection
PullRequestChangedFileEdge
PullRequestCommit
PullRequestCommitCommentThread
PullRequestCommitConnection

PullRequestCommitEdge
PullRequestConnection
PullRequestContributionsByRepository
PullRequestEdge
PullRequestParameters
PullRequestReview
PullRequestReviewComment
PullRequestReviewCommentConnection
PullRequestReviewCommentEdge
PullRequestReviewConnection
PullRequestReviewContributionsByRepository
PullRequestReviewEdge
PullRequestReviewThread
PullRequestReviewThreadConnection
PullRequestReviewThreadEdge
PullRequestRevisionMarker
PullRequestTemplate
PullRequestThread
PullRequestTimelineConnection
PullRequestTimelineItemEdge
PullRequestTimelineItemsConnection
PullRequestTimelineItemsEdge
Push
PushAllowance
PushAllowanceConnection
PushAllowanceEdge
RateLimit
ReactingUserConnection
ReactingUserEdge
Reaction
ReactionConnection
ReactionEdge
ReactionGroup
ReactorConnection
ReactorEdge
ReadyForReviewEvent
Ref
RefConnection
RefEdge
RefNameConditionTarget
RefUpdateRule
ReferencedEvent
Release
ReleaseAsset
ReleaseAssetConnection
ReleaseAssetEdge
ReleaseConnection
ReleaseEdge
RemovedFromMergeQueueEvent
RemovedFromProjectEvent
RenamedTitleEvent

ReopenedEvent
RepoAccessAuditEntry
RepoAddMemberAuditEntry
RepoAddTopicAuditEntry
RepoArchivedAuditEntry
RepoChangeMergeSettingAuditEntry
RepoConfigDisableAnonymousGitAccessAuditEntry
RepoConfigDisableCollaboratorsOnlyAuditEntry
RepoConfigDisableContributorsOnlyAuditEntry
RepoConfigDisableSockpuppetDisallowedAuditEntry
RepoConfigEnableAnonymousGitAccessAuditEntry
RepoConfigEnableCollaboratorsOnlyAuditEntry
RepoConfigEnableContributorsOnlyAuditEntry
RepoConfigEnableSockpuppetDisallowedAuditEntry
RepoConfigLockAnonymousGitAccessAuditEntry
RepoConfigUnlockAnonymousGitAccessAuditEntry
RepoCreateAuditEntry
RepoDestroyAuditEntry
RepoRemoveMemberAuditEntry
RepoRemoveTopicAuditEntry
Repository
RepositoryCodeowners
RepositoryCodeownersError
RepositoryCollaboratorConnection
RepositoryCollaboratorEdge
RepositoryConnection
RepositoryContactLink
RepositoryEdge
RepositoryIdConditionTarget
RepositoryInteractionAbility
RepositoryInvitation
RepositoryInvitationConnection
RepositoryInvitationEdge
RepositoryMigration
RepositoryMigrationConnection
RepositoryMigrationEdge
RepositoryNameConditionTarget
RepositoryRule
RepositoryRuleConditions
RepositoryRuleConnection
RepositoryRuleEdge
RepositoryRuleset
RepositoryRulesetBypassActor
RepositoryRulesetBypassActorConnection
RepositoryRulesetBypassActorEdge
RepositoryRulesetConnection
RepositoryRulesetEdge
RepositoryTopic
RepositoryTopicConnection
RepositoryTopicEdge
RepositoryVisibilityChangeDisableAuditEntry

RepositoryVisibilityChangeEnableAuditEntry
RepositoryVulnerabilityAlert
RepositoryVulnerabilityAlertConnection
RepositoryVulnerabilityAlertEdge
RequestedReviewerConnection
RequestedReviewerEdge
RequiredDeploymentsParameters
RequiredStatusCheckDescription
RequiredStatusChecksParameters
RestrictedContribution
ReviewDismissalAllowance
ReviewDismissalAllowanceConnection
ReviewDismissalAllowanceEdge
ReviewDismissedEvent
ReviewRequest
ReviewRequestConnection
ReviewRequestEdge
ReviewRequestRemovedEvent
ReviewRequestedEvent
ReviewStatusHovercardContext
SavedReply
SavedReplyConnection
SavedReplyEdge
SearchResultItemConnection
SearchResultItemEdge
SecurityAdvisory
SecurityAdvisoryConnection
SecurityAdvisoryEdge
SecurityAdvisoryIdentifier
SecurityAdvisoryPackage
SecurityAdvisoryPackageVersion
SecurityAdvisoryReference
SecurityVulnerability
SecurityVulnerabilityConnection
SecurityVulnerabilityEdge
SmimeSignature
SocialAccount
SocialAccountConnection
SocialAccountEdge
SponsorConnection
SponsorEdge
SponsorableItemConnection
SponsorableItemEdge
SponsorsActivity
SponsorsActivityConnection
SponsorsActivityEdge
SponsorsGoal
SponsorsListing
SponsorsListingFeaturedItem
SponsorsTier
SponsorsTierAdminInfo

SponsorsTierAdminInfo
SponsorsTierConnection
SponsorsTierEdge
Sponsorship
SponsorshipConnection
SponsorshipEdge
SponsorshipNewsletter
SponsorshipNewsletterConnection
SponsorshipNewsletterEdge
SshSignature
StargazerConnection
StargazerEdge
StarredRepositoryConnection
StarredRepositoryEdge
Status
StatusCheckConfiguration
StatusCheckRollup
StatusCheckRollupContextConnection
StatusCheckRollupContextEdge
StatusContext
StatusContextStateCount
StripeConnectAccount
Submodule
SubmoduleConnection
SubmoduleEdge
SubscribedEvent
SuggestedReviewer
Tag
TagNamePatternParameters
Team
TeamAddMemberAuditEntry
TeamAddRepositoryAuditEntry
TeamChangeParentTeamAuditEntry
TeamConnection
TeamDiscussion
TeamDiscussionComment
TeamDiscussionCommentConnection
TeamDiscussionCommentEdge
TeamDiscussionConnection
TeamDiscussionEdge
TeamEdge
TeamMemberConnection
TeamMemberEdge
TeamRemoveMemberAuditEntry
TeamRemoveRepositoryAuditEntry
TeamRepositoryConnection
TeamRepositoryEdge
TextMatch
TextMatchHighlight
Topic
TransferredEvent
Tree

Tree
TreeEntry
UnassignedEvent
UnknownSignature
UnlabeledEvent
UnlockedEvent
UnmarkedAsDuplicateEvent
UnpinnedEvent
UnsubscribedEvent
UpdateParameters
User
UserBlockedEvent
UserConnection
UserContentEdit
UserContentEditConnection
UserContentEditEdge
UserEdge
UserEmailMetadata
UserStatus
UserStatusConnection
UserStatusEdge
VerifiableDomain
VerifiableDomainConnection
VerifiableDomainEdge
ViewerHovercardContext
Workflow
WorkflowFileReference
WorkflowRun
WorkflowRunConnection
WorkflowRunEdge
WorkflowRunFile
WorkflowsParameters

Objects in GraphQL represent the resources you can access.

About objects

Objects in GraphQL represent the resources you can access. An object can contain a list of fields, which are specifically typed.

For example, the `Repository` object has a field called `name`, which is a `String`.

For more information, see "Introduction to GraphQL."

ActorLocation

Location information for an actor.

Fields for `ActorLocation`

Name	Description
`city` (`String`)	City.
`country` (`String`)	Country name.
`countryCode` (`String`)	Country code.
`region` (`String`)	Region name.
`regionCode` (`String`)	Region or state code.

AddedToMergeQueueEvent

Represents an `added_to_merge_queue` event on a given pull request.

`AddedToMergeQueueEvent` Implements

- `Node`

Fields for `AddedToMergeQueueEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`enqueuer` (`User`)	The user who added this Pull Request to the merge queue.
`mergeQueue` (`MergeQueue`)	The merge queue where this pull request was added to.
`pullRequest` (`PullRequest`)	PullRequest referenced by event.

AddedToProjectEvent

Represents a `added_to_project` event on a given issue or pull request.

`AddedToProjectEvent` Implements

- `Node`

Fields for `AddedToProjectEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.

`databaseId` (`Int`)	Identifies the primary key from the database.
`project` (`Project`)	Project referenced by event. **Preview notice** `project` is available under the Project event details preview. During the preview period, the API may change without notice.
`projectCard` (`ProjectCard`)	Project card referenced by this project event. **Preview notice** `projectCard` is available under the Project event details preview. During the preview period, the API may change without notice.
`projectColumnName` (`String!`)	Column name referenced by this project event. **Preview notice** `projectColumnName` is available under the Project event details preview. During the preview period, the API may change without notice.

App

A GitHub App.

`App` Implements

- `Node`

Fields for `App`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`description` (`String`)	The description of the app.
`ipAllowListEntries` (`IpAllowListEntryConnection!`)	The IP addresses of the app. **Arguments for `ipAllowListEntries`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come

Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`IpAllowListEntryOrder`)
Ordering options for IP allow list entries returned.

`logoBackgroundColor` (`String!`)	The hex color code, without the leading '#', for the logo background.
`logoUrl` (`URI!`)	A URL pointing to the app's logo. **Arguments for `logoUrl`** `size` (`Int`) The size of the resulting image.
`name` (`String!`)	The name of the app.
`slug` (`String!`)	A slug based on the name of the app for use in URLs.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The URL to the app's homepage.

AssignedEvent

Represents an `assigned` event on any assignable object.

`AssignedEvent` Implements

- `Node`

Fields for `AssignedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`assignable` (`Assignable!`)	Identifies the assignable associated with the event.
`assignee` (`Assignee`)	Identifies the user or mannequin that was assigned.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.

`user` (`User`)	Identifies the user who was assigned. **Deprecation notice** `user` is deprecated. Assignees can now be mannequins. Use the `assignee` field instead. Removal on 2020-01-01 UTC.

AutoMergeDisabledEvent

Represents a `auto_merge_disabled` event on a given pull request.

`AutoMergeDisabledEvent` Implements

- `Node`

Fields for `AutoMergeDisabledEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`disabler` (`User`)	The user who disabled auto-merge for this Pull Request.
`pullRequest` (`PullRequest`)	PullRequest referenced by event.
`reason` (`String`)	The reason auto-merge was disabled.
`reasonCode` (`String`)	The reason_code relating to why auto-merge was disabled.

AutoMergeEnabledEvent

Represents a `auto_merge_enabled` event on a given pull request.

`AutoMergeEnabledEvent` Implements

- `Node`

Fields for `AutoMergeEnabledEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`enabler` (`User`)	The user who enabled auto-merge for this Pull Request.

`pullRequest` (`PullRequest`)	PullRequest referenced by event.

AutoMergeRequest

Represents an auto-merge request for a pull request.

Fields for `AutoMergeRequest`

Name	Description
`authorEmail` (`String`)	The email address of the author of this auto-merge request.
`commitBody` (`String`)	The commit message of the auto-merge request. If a merge queue is required by the base branch, this value will be set by the merge queue when merging.
`commitHeadline` (`String`)	The commit title of the auto-merge request. If a merge queue is required by the base branch, this value will be set by the merge queue when merging.
`enabledAt` (`DateTime`)	When was this auto-merge request was enabled.
`enabledBy` (`Actor`)	The actor who created the auto-merge request.
`mergeMethod` (`PullRequestMergeMethod!`)	The merge method of the auto-merge request. If a merge queue is required by the base branch, this value will be set by the merge queue when merging.
`pullRequest` (`PullRequest!`)	The pull request that this auto-merge request is set against.

AutoRebaseEnabledEvent

Represents a `auto_rebase_enabled` event on a given pull request.

`AutoRebaseEnabledEvent` Implements

- `Node`

Fields for `AutoRebaseEnabledEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`enabler` (`User`)	The user who enabled auto-merge (rebase) for this Pull Request.
`pullRequest` (`PullRequest`)	PullRequest referenced by event.

AutoSquashEnabledEvent

Represents a `auto_squash_enabled` event on a given pull request.

`AutoSquashEnabledEvent` Implements

- `Node`

Fields for `AutoSquashEnabledEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`enabler` (`User`)	The user who enabled auto-merge (squash) for this Pull Request.
`pullRequest` (`PullRequest`)	PullRequest referenced by event.

AutomaticBaseChangeFailedEvent

Represents a `automatic_base_change_failed` event on a given pull request.

`AutomaticBaseChangeFailedEvent` Implements

- `Node`

Fields for `AutomaticBaseChangeFailedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`newBase` (`String!`)	The new base for this PR.
`oldBase` (`String!`)	The old base for this PR.
`pullRequest` (`PullRequest!`)	PullRequest referenced by event.

AutomaticBaseChangeSucceededEvent

Represents a `automatic_base_change_succeeded` event on a given pull request.

`AutomaticBaseChangeSucceededEvent` Implements

- `Node`

Fields for `AutomaticBaseChangeSucceededEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`newBase` (`String!`)	The new base for this PR.
`oldBase` (`String!`)	The old base for this PR.
`pullRequest` (`PullRequest!`)	PullRequest referenced by event.

BaseRefChangedEvent

Represents a `base_ref_changed` event on a given issue or pull request.

`BaseRefChangedEvent` Implements

- `Node`

Fields for `BaseRefChangedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`currentRefName` (`String!`)	Identifies the name of the base ref for the pull request after it was changed.
`databaseId` (`Int`)	Identifies the primary key from the database.
`previousRefName` (`String!`)	Identifies the name of the base ref for the pull request before it was changed.
`pullRequest` (`PullRequest!`)	PullRequest referenced by event.

BaseRefDeletedEvent

Represents a `base_ref_deleted` event on a given pull request.

`BaseRefDeletedEvent` Implements

- `Node`

Fields for `BaseRefDeletedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.

`baseRefName` (`String`)	Identifies the name of the Ref associated with the `base_ref_deleted` event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`pullRequest` (`PullRequest`)	PullRequest referenced by event.

BaseRefForcePushedEvent

Represents a `base_ref_force_pushed` event on a given pull request.

`BaseRefForcePushedEvent` Implements

- `Node`

Fields for `BaseRefForcePushedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`afterCommit` (`Commit`)	Identifies the after commit SHA for the `base_ref_force_pushed` event.
`beforeCommit` (`Commit`)	Identifies the before commit SHA for the `base_ref_force_pushed` event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`pullRequest` (`PullRequest!`)	PullRequest referenced by event.
`ref` (`Ref`)	Identifies the fully qualified ref name for the `base_ref_force_pushed` event.

Blame

Represents a Git blame.

Fields for `Blame`

Name	Description
`ranges` (`[BlameRange!]!`)	The list of ranges from a Git blame.

BlameRange

Represents a range of information from a Git blame.

Fields for `BlameRange`

Name	Description

`age` (`Int!`)	Identifies the recency of the change, from 1 (new) to 10 (old). This is calculated as a 2-quantile and determines the length of distance between the median age of all the changes in the file and the recency of the current range's change.
`commit` (`Commit!`)	Identifies the line author.
`endingLine` (`Int!`)	The ending line for the range.
`startingLine` (`Int!`)	The starting line for the range.

Blob

Represents a Git blob.

`Blob` Implements

- `GitObject`
- `Node`

Fields for `Blob`

Name	Description
`abbreviatedOid` (`String!`)	An abbreviated version of the Git object ID.
`byteSize` (`Int!`)	Byte size of Blob object.
`commitResourcePath` (`URI!`)	The HTTP path for this Git object.
`commitUrl` (`URI!`)	The HTTP URL for this Git object.
`isBinary` (`Boolean`)	Indicates whether the Blob is binary or text. Returns null if unable to determine the encoding.
`isTruncated` (`Boolean!`)	Indicates whether the contents is truncated.
`oid` (`GitObjectID!`)	The Git object ID.
`repository` (`Repository!`)	The Repository the Git object belongs to.
`text` (`String`)	UTF8 text data or null if the Blob is binary.

Bot

A special type of user which takes actions on behalf of GitHub Apps.

`Bot` Implements

- `Actor`
- `Node`
- `UniformResourceLocatable`

Fields for `Bot`

Name	Description
`avatarUrl` (`URI!`)	A URL pointing to the GitHub App's public avatar. **Arguments for `avatarUrl`** `size` (`Int`) The size of the resulting square image.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`login` (`String!`)	The username of the actor.
`resourcePath` (`URI!`)	The HTTP path for this bot.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this bot.

BranchNamePatternParameters

Parameters to be used for the branch_name_pattern rule.

Fields for `BranchNamePatternParameters`

Name	Description
`name` (`String`)	How this rule will appear to users.
`negate` (`Boolean!`)	If true, the rule will fail if the pattern matches.
`operator` (`String!`)	The operator to use for matching.
`pattern` (`String!`)	The pattern to match with.

BranchProtectionRule

A branch protection rule.

`BranchProtectionRule` Implements

- `Node`

Fields for `BranchProtectionRule`

Name	Description

`allowsDeletions` (`Boolean!`)	Can this branch be deleted.
`allowsForcePushes` (`Boolean!`)	Are force pushes allowed on this branch.
`blocksCreations` (`Boolean!`)	Is branch creation a protected operation.
`branchProtectionRuleConflicts` (`BranchProtectionRuleConflictConnection!`)	A list of conflicts matching branches protection rule and other branch protection rules. **Arguments for `branchProtectionRuleConflicts`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`bypassForcePushAllowances` (`BypassForcePushAllowanceConnection!`)	A list of actors able to force push for this branch protection rule. **Arguments for `bypassForcePushAllowances`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`bypassPullRequestAllowances` (`BypassPullRequestAllowanceConnection!`)	A list of actors able to bypass PRs for this branch protection rule. **Arguments for `bypassPullRequestAllowances`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come

before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`creator` (`Actor`)	The actor who created this branch protection rule.
`databaseId` (`Int`)	Identifies the primary key from the database.
`dismissesStaleReviews` (`Boolean!`)	Will new commits pushed to matching branches dismiss pull request review approvals.
`isAdminEnforced` (`Boolean!`)	Can admins overwrite branch protection.
`lockAllowsFetchAndMerge` (`Boolean!`)	Whether users can pull changes from upstream when the branch is locked. Set to `true` to allow fork syncing. Set to `false` to prevent fork syncing.
`lockBranch` (`Boolean!`)	Whether to set the branch as read-only. If this is true, users will not be able to push to the branch.
`matchingRefs` (`RefConnection!`)	Repository refs that are protected by this rule.

Arguments for `matchingRefs`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`query` (`String`)
Filters refs with query on name.

`pattern` (`String!`)	Identifies the protection rule pattern.
`pushAllowances` (`PushAllowanceConnection!`)	A list push allowances for this branch protection rule.

Arguments for `pushAllowances`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`repository` (`Repository`)	The repository associated with this branch protection rule.
`requireLastPushApproval` (`Boolean!`)	Whether the most recent push must be approved by someone other than the person who pushed it.
`requiredApprovingReviewCount` (`Int`)	Number of approving reviews required to update matching branches.
`requiredDeploymentEnvironments` (`[String]`)	List of required deployment environments that must be deployed successfully to update matching branches.
`requiredStatusCheckContexts` (`[String]`)	List of required status check contexts that must pass for commits to be accepted to matching branches.
`requiredStatusChecks` (`[RequiredStatusCheckDescription!]`)	List of required status checks that must pass for commits to be accepted to matching branches.
`requiresApprovingReviews` (`Boolean!`)	Are approving reviews required to update matching branches.
`requiresCodeOwnerReviews` (`Boolean!`)	Are reviews from code owners required to update matching branches.
`requiresCommitSignatures` (`Boolean!`)	Are commits required to be signed.
`requiresConversationResolution` (`Boolean!`)	Are conversations required to be resolved before merging.
`requiresDeployments` (`Boolean!`)	Does this branch require deployment to specific environments before merging.
`requiresLinearHistory` (`Boolean!`)	Are merge commits prohibited from being pushed to this branch.
`requiresStatusChecks` (`Boolean!`)	Are status checks required to update matching branches.
`requiresStrictStatusChecks` (`Boolean!`)	Are branches required to be up to date before merging.
`restrictsPushes` (`Boolean!`)	Is pushing to matching branches restricted.
`restrictsReviewDismissals` (`Boolean!`)	Is dismissal of pull request reviews restricted.
`reviewDismissalAllowances` (`ReviewDismissalAllowanceConnection!`)	A list review dismissal allowances for this branch protection rule.

Arguments for `reviewDismissalAllowances`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

BranchProtectionRuleConflict

A conflict between two branch protection rules.

Fields for `BranchProtectionRuleConflict`

Name	Description
`branchProtectionRule` (`BranchProtectionRule`)	Identifies the branch protection rule.
`conflictingBranchProtectionRule` (`BranchProtectionRule`)	Identifies the conflicting branch protection rule.
`ref` (`Ref`)	Identifies the branch ref that has conflicting rules.

BranchProtectionRuleConflictConnection

The connection type for BranchProtectionRuleConflict.

Fields for `BranchProtectionRuleConflictConnection`

Name	Description
`edges` (`[BranchProtectionRuleConflictEdge]`)	A list of edges.
`nodes` (`[BranchProtectionRuleConflict]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

BranchProtectionRuleConflictEdge

An edge in a connection.

Fields for `BranchProtectionRuleConflictEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`BranchProtectionRuleConflict`)	The item at the end of the edge.

BranchProtectionRuleConnection

The connection type for BranchProtectionRule.

Fields for `BranchProtectionRuleConnection`

Name	Description
`edges` (`[BranchProtectionRuleEdge]`)	A list of edges.
`nodes` (`[BranchProtectionRule]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

BranchProtectionRuleEdge

An edge in a connection.

Fields for `BranchProtectionRuleEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`BranchProtectionRule`)	The item at the end of the edge.

BypassForcePushAllowance

A user, team, or app who has the ability to bypass a force push requirement on a protected branch.

`BypassForcePushAllowance` Implements

- `Node`

Fields for `BypassForcePushAllowance`

Name	Description
`actor` (`BranchActorAllowanceActor`)	The actor that can force push.
`branchProtectionRule` (`BranchProtectionRule`)	Identifies the branch protection rule associated

	with the allowed user, team, or app.

BypassForcePushAllowanceConnection

The connection type for BypassForcePushAllowance.

Fields for `BypassForcePushAllowanceConnection`

Name	Description
`edges` (`[BypassForcePushAllowanceEdge]`)	A list of edges.
`nodes` (`[BypassForcePushAllowance]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

BypassForcePushAllowanceEdge

An edge in a connection.

Fields for `BypassForcePushAllowanceEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`BypassForcePushAllowance`)	The item at the end of the edge.

BypassPullRequestAllowance

A user, team, or app who has the ability to bypass a pull request requirement on a protected branch.

`BypassPullRequestAllowance` Implements

- `Node`

Fields for `BypassPullRequestAllowance`

Name	Description
`actor` (`BranchActorAllowanceActor`)	The actor that can bypass.
`branchProtectionRule` (`BranchProtectionRule`)	Identifies the branch protection rule associated with the allowed user, team, or app.

BypassPullRequestAllowanceConnection

The connection type for BypassPullRequestAllowance.

Fields for `BypassPullRequestAllowanceConnection`

Name	Description
`edges` (`[BypassPullRequestAllowanceEdge]`)	A list of edges.
`nodes` (`[BypassPullRequestAllowance]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

BypassPullRequestAllowanceEdge

An edge in a connection.

Fields for `BypassPullRequestAllowanceEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`BypassPullRequestAllowance`)	The item at the end of the edge.

CVSS

The Common Vulnerability Scoring System.

Fields for `CVSS`

Name	Description
`score` (`Float!`)	The CVSS score associated with this advisory.
`vectorString` (`String`)	The CVSS vector string associated with this advisory.

CWE

A common weakness enumeration.

`CWE` Implements

- `Node`

Fields for `CWE`

Name	Description
`cweId` (`String!`)	The id of the CWE.

`description` (`String!`)	A detailed description of this CWE.
`name` (`String!`)	The name of this CWE.

CWEConnection

The connection type for CWE.

Fields for `CWEConnection`

Name	Description
`edges` (`[CWEEdge]`)	A list of edges.
`nodes` (`[CWE]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

CWEEdge

An edge in a connection.

Fields for `CWEEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`CWE`)	The item at the end of the edge.

CheckAnnotation

A single check annotation.

Fields for `CheckAnnotation`

Name	Description
`annotationLevel` (`CheckAnnotationLevel`)	The annotation's severity level.
`blobUrl` (`URI!`)	The path to the file that this annotation was made on.
`databaseId` (`Int`)	Identifies the primary key from the database.
`location` (`CheckAnnotationSpan!`)	The position of this annotation.
`message` (`String!`)	The annotation's message.

`path` (`String!`)	The path that this annotation was made on.
`rawDetails` (`String`)	Additional information about the annotation.
`title` (`String`)	The annotation's title.

CheckAnnotationConnection

The connection type for CheckAnnotation.

Fields for `CheckAnnotationConnection`

Name	Description
`edges` (`[CheckAnnotationEdge]`)	A list of edges.
`nodes` (`[CheckAnnotation]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

CheckAnnotationEdge

An edge in a connection.

Fields for `CheckAnnotationEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`CheckAnnotation`)	The item at the end of the edge.

CheckAnnotationPosition

A character position in a check annotation.

Fields for `CheckAnnotationPosition`

Name	Description
`column` (`Int`)	Column number (1 indexed).
`line` (`Int!`)	Line number (1 indexed).

CheckAnnotationSpan

An inclusive pair of positions for a check annotation.

Fields for `CheckAnnotationSpan`

Name	Description
`end` (`CheckAnnotationPosition!`)	End position (inclusive).
`start` (`CheckAnnotationPosition!`)	Start position (inclusive).

CheckRun

A check run.

`CheckRun` Implements

- `Node`
- `RequirableByPullRequest`
- `UniformResourceLocatable`

Fields for `CheckRun`

Name	Description
`annotations` (`CheckAnnotationConnection`)	The check run's annotations. **Arguments for `annotations`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`checkSuite` (`CheckSuite!`)	The check suite that this run is a part of.
`completedAt` (`DateTime`)	Identifies the date and time when the check run was completed.
`conclusion` (`CheckConclusionState`)	The conclusion of the check run.
`databaseId` (`Int`)	Identifies the primary key from the database.
`deployment` (`Deployment`)	The corresponding deployment for this job, if any.
`detailsUrl` (`URI`)	The URL from which to find full details of the check run on the integrator's site.
`externalId` (`String`)	A reference for the check run on the integrator's

	system.
`isRequired` (`Boolean!`)	Whether this is required to pass before merging for a specific pull request. **Arguments for `isRequired`** `pullRequestId` (`ID`) The id of the pull request this is required for. `pullRequestNumber` (`Int`) The number of the pull request this is required for.
`name` (`String!`)	The name of the check for this check run.
`pendingDeploymentRequest` (`DeploymentRequest`)	Information about a pending deployment, if any, in this check run.
`permalink` (`URI!`)	The permalink to the check run summary.
`repository` (`Repository!`)	The repository associated with this check run.
`resourcePath` (`URI!`)	The HTTP path for this check run.
`startedAt` (`DateTime`)	Identifies the date and time when the check run was started.
`status` (`CheckStatusState!`)	The current status of the check run.
`steps` (`CheckStepConnection`)	The check run's steps. **Arguments for `steps`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `number` (`Int`) Step number.
`summary` (`String`)	A string representing the check run's summary.
`text` (`String`)	A string representing the check run's text.
`title` (`String`)	A string representing the check run.

`url` (`URI!`)	The HTTP URL for this check run.

CheckRunConnection

The connection type for CheckRun.

Fields for `CheckRunConnection`

Name	Description
`edges` (`[CheckRunEdge]`)	A list of edges.
`nodes` (`[CheckRun]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

CheckRunEdge

An edge in a connection.

Fields for `CheckRunEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`CheckRun`)	The item at the end of the edge.

CheckRunStateCount

Represents a count of the state of a check run.

Fields for `CheckRunStateCount`

Name	Description
`count` (`Int!`)	The number of check runs with this state.
`state` (`CheckRunState!`)	The state of a check run.

CheckStep

A single check step.

Fields for `CheckStep`

Name	Description
`completedAt` (`DateTime`)	Identifies the date and time when the check

`completedAt` (`DateTime`)	Identifies the date and time when the check step was completed.
`conclusion` (`CheckConclusionState`)	The conclusion of the check step.
`externalId` (`String`)	A reference for the check step on the integrator's system.
`name` (`String!`)	The step's name.
`number` (`Int!`)	The index of the step in the list of steps of the parent check run.
`secondsToCompletion` (`Int`)	Number of seconds to completion.
`startedAt` (`DateTime`)	Identifies the date and time when the check step was started.
`status` (`CheckStatusState!`)	The current status of the check step.

CheckStepConnection

The connection type for CheckStep.

Fields for `CheckStepConnection`

Name	Description
`edges` (`[CheckStepEdge]`)	A list of edges.
`nodes` (`[CheckStep]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

CheckStepEdge

An edge in a connection.

Fields for `CheckStepEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`CheckStep`)	The item at the end of the edge.

CheckSuite

A check suite.

`CheckSuite` Implements

- Node

Fields for `CheckSuite`

Name	Description
`app` (App)	The GitHub App which created this check suite.
`branch` (Ref)	The name of the branch for this check suite.
`checkRuns` (CheckRunConnection)	The check runs associated with a check suite. **Arguments for `checkRuns`** `after` (String) Returns the elements in the list that come after the specified cursor. `before` (String) Returns the elements in the list that come before the specified cursor. `filterBy` (CheckRunFilter) Filters the check runs by this type. `first` (Int) Returns the first *n* elements from the list. `last` (Int) Returns the last *n* elements from the list.
`commit` (Commit!)	The commit for this check suite.
`conclusion` (CheckConclusionState)	The conclusion of this check suite.
`createdAt` (DateTime!)	Identifies the date and time when the object was created.
`creator` (User)	The user who triggered the check suite.
`databaseId` (Int)	Identifies the primary key from the database.
`matchingPullRequests` (PullRequestConnection)	A list of open pull requests matching the check suite. **Arguments for `matchingPullRequests`** `after` (String) Returns the elements in the list that come after the specified cursor. `baseRefName` (String) The base ref name to filter the pull requests by. `before` (String) Returns the elements in the list that come before the specified cursor. `first` (Int)

Returns the first *n* elements from the list.

`headRefName` (`String`)
The head ref name to filter the pull requests by.

`labels` (`[String!]`)
A list of label names to filter the pull requests by.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`IssueOrder`)
Ordering options for pull requests returned from the connection.

`states` (`[PullRequestState!]`)
A list of states to filter the pull requests by.

`push` (`Push`)	The push that triggered this check suite.
`repository` (`Repository!`)	The repository associated with this check suite.
`resourcePath` (`URI!`)	The HTTP path for this check suite.
`status` (`CheckStatusState!`)	The status of this check suite.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this check suite.
`workflowRun` (`WorkflowRun`)	The workflow run associated with this check suite.

CheckSuiteConnection

The connection type for CheckSuite.

Fields for `CheckSuiteConnection`

Name	Description
`edges` (`[CheckSuiteEdge]`)	A list of edges.
`nodes` (`[CheckSuite]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

CheckSuiteEdge

An edge in a connection.

Fields for `CheckSuiteEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`CheckSuite`)	The item at the end of the edge.

ClosedEvent

Represents a `closed` event on any `Closable`.

`ClosedEvent` Implements

- `Node`
- `UniformResourceLocatable`

Fields for `ClosedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`closable` (`Closable!`)	Object that was closed.
`closer` (`Closer`)	Object which triggered the creation of this event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`resourcePath` (`URI!`)	The HTTP path for this closed event.
`stateReason` (`IssueStateReason`)	The reason the issue state was changed to closed.
`url` (`URI!`)	The HTTP URL for this closed event.

CodeOfConduct

The Code of Conduct for a repository.

`CodeOfConduct` Implements

- `Node`

Fields for `CodeOfConduct`

Name	Description
`body` (`String`)	The body of the Code of Conduct.
`key` (`String!`)	The key for the Code of Conduct.
`name` (`String!`)	The formal name of the Code of Conduct.

`resourcePath` (`URI`)	The HTTP path for this Code of Conduct.
`url` (`URI`)	The HTTP URL for this Code of Conduct.

CommentDeletedEvent

Represents a `comment_deleted` event on a given issue or pull request.

`CommentDeletedEvent` Implements

- `Node`

Fields for `CommentDeletedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`deletedCommentAuthor` (`Actor`)	The user who authored the deleted comment.

Commit

Represents a Git commit.

`Commit` Implements

- `GitObject`
- `Node`
- `Subscribable`
- `UniformResourceLocatable`

Fields for `Commit`

Name	Description
`abbreviatedOid` (`String!`)	An abbreviated version of the Git object ID.
`additions` (`Int!`)	The number of additions in this commit.
`associatedPullRequests` (`PullRequestConnection`)	The merged Pull Request that introduced the commit to the repository. If the commit is not present in the default branch, additionally returns open Pull Requests associated with the commit. **Arguments for `associatedPullRequests`** `after` (`String`) Returns the elements in the list that come

Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`PullRequestOrder`)
Ordering options for pull requests.

`author` (`GitActor`)	Authorship details of the commit.
`authoredByCommitter` (`Boolean!`)	Check if the committer and the author match.
`authoredDate` (`DateTime!`)	The datetime when this commit was authored.
`authors` (`GitActorConnection!`)	The list of authors for this commit based on the git author and the Co-authored-by message trailer. The git author will always be first.

Arguments for `authors`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`blame` (`Blame!`)	Fetches `git blame` information.

Arguments for `blame`

`path` (`String!`)
The file whose Git blame information you want.

`changedFiles` (`Int!`)	We recommend using the `changedFilesIfAvailable` field instead of `changedFiles`, as `changedFiles` will cause your request to return an error if GitHub is unable to calculate the number of changed files.

Deprecation notice
`changedFiles` is deprecated.

`changedFiles` will be removed. Use `changedFilesIfAvailable` instead. Removal on 2023-01-01 UTC.

`changedFilesIfAvailable` (`Int`)	The number of changed files in this commit. If GitHub is unable to calculate the number of changed files (for example due to a timeout), this will return `null`. We recommend using this field instead of `changedFiles`.
`checkSuites` (`CheckSuiteConnection`)	The check suites associated with a commit. **Arguments for `checkSuites`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `filterBy` (`CheckSuiteFilter`) Filters the check suites by this type. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`comments` (`CommitCommentConnection!`)	Comments made on the commit. **Arguments for `comments`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`commitResourcePath` (`URI!`)	The HTTP path for this Git object.
`commitUrl` (`URI!`)	The HTTP URL for this Git object.
`committedDate` (`DateTime!`)	The datetime when this commit was committed.
`committedViaWeb` (`Boolean!`)	Check if committed via GitHub web UI.

`committer` (`GitActor`)	Committer details of the commit.
`deletions` (`Int!`)	The number of deletions in this commit.
`deployments` (`DeploymentConnection`)	The deployments associated with a commit. **Arguments for `deployments`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `environments` (`[String!]`) Environments to list deployments for. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`DeploymentOrder`) Ordering options for deployments returned from the connection.
`file` (`TreeEntry`)	The tree entry representing the file located at the given path. **Arguments for `file`** `path` (`String!`) The path for the file.
`history` (`CommitHistoryConnection!`)	The linear commit history starting from (and including) this commit, in the same order as `git log` . **Arguments for `history`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `author` (`CommitAuthor`) If non-null, filters history to only show commits with matching authorship. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.

`path` (`String`)
If non-null, filters history to only show commits touching files under this path.

`since` (`GitTimestamp`)
Allows specifying a beginning time or date for fetching commits.

`until` (`GitTimestamp`)
Allows specifying an ending time or date for fetching commits.

`message` (`String!`)	The Git commit message.
`messageBody` (`String!`)	The Git commit message body.
`messageBodyHTML` (`HTML!`)	The commit message body rendered to HTML.
`messageHeadline` (`String!`)	The Git commit message headline.
`messageHeadlineHTML` (`HTML!`)	The commit message headline rendered to HTML.
`oid` (`GitObjectID!`)	The Git object ID.
`onBehalfOf` (`Organization`)	The organization this commit was made on behalf of.
`parents` (`CommitConnection!`)	The parents of a commit.

Arguments for `parents`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`pushedDate` (`DateTime`)	The datetime when this commit was pushed.

Deprecation notice
`pushedDate` is deprecated.
`pushedDate` is no longer supported.
Removal on 2023-07-01 UTC.

`repository` (`Repository!`)	The Repository this commit belongs to.
`resourcePath` (`URI!`)	The HTTP path for this commit.

`signature` (`GitSignature`)	Commit signing information, if present.
`status` (`Status`)	Status information for this commit.
`statusCheckRollup` (`StatusCheckRollup`)	Check and Status rollup information for this commit.
`submodules` (`SubmoduleConnection!`)	Returns a list of all submodules in this repository as of this Commit parsed from the .gitmodules file. **Arguments for `submodules`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`tarballUrl` (`URI!`)	Returns a URL to download a tarball archive for a repository. Note: For private repositories, these links are temporary and expire after five minutes.
`tree` (`Tree!`)	Commit's root Tree.
`treeResourcePath` (`URI!`)	The HTTP path for the tree of this commit.
`treeUrl` (`URI!`)	The HTTP URL for the tree of this commit.
`url` (`URI!`)	The HTTP URL for this commit.
`viewerCanSubscribe` (`Boolean!`)	Check if the viewer is able to change their subscription status for the repository.
`viewerSubscription` (`SubscriptionState`)	Identifies if the viewer is watching, not watching, or ignoring the subscribable entity.
`zipballUrl` (`URI!`)	Returns a URL to download a zipball archive for a repository. Note: For private repositories, these links are temporary and expire after five minutes.

CommitAuthorEmailPatternParameters

Parameters to be used for the commit_author_email_pattern rule.

Fields for `CommitAuthorEmailPatternParameters`

Name	Description
`name` (`String`)	How this rule will appear to users.
`negate` (`Boolean!`)	If true, the rule will fail if the pattern matches.
`operator` (`String!`)	The operator to use for matching.
`pattern` (`String!`)	The pattern to match with.

CommitComment

Represents a comment on a given Commit.

`CommitComment` Implements

- `Comment`
- `Deletable`
- `Minimizable`
- `Node`
- `Reactable`
- `RepositoryNode`
- `Updatable`
- `UpdatableComment`

Fields for `CommitComment`

Name	Description
`author` (`Actor`)	The actor who authored the comment.
`authorAssociation` (`CommentAuthorAssociation!`)	Author's association with the subject of the comment.
`body` (`String!`)	Identifies the comment body.
`bodyHTML` (`HTML!`)	The body rendered to HTML.
`bodyText` (`String!`)	The body rendered to text.
`commit` (`Commit`)	Identifies the commit associated with the comment, if the commit exists.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`createdViaEmail` (`Boolean!`)	Check if this comment was created via an email reply.
`databaseId` (`Int`)	Identifies the primary key from the database.
`editor` (`Actor`)	The actor who edited the comment.
`includesCreatedEdit` (`Boolean!`)	Check if this comment was edited and includes an edit with the creation data.
`isMinimized` (`Boolean!`)	Returns whether or not a comment has been minimized.

`lastEditedAt` (`DateTime`)	The moment the editor made the last edit.
`minimizedReason` (`String`)	Returns why the comment was minimized. One of `abuse`, `off-topic`, `outdated`, `resolved`, `duplicate` and `spam`. Note that the case and formatting of these values differs from the inputs to the `MinimizeComment` mutation.
`path` (`String`)	Identifies the file path associated with the comment.
`position` (`Int`)	Identifies the line position associated with the comment.
`publishedAt` (`DateTime`)	Identifies when the comment was published at.
`reactionGroups` (`[ReactionGroup!]`)	A list of reactions grouped by content left on the subject.
`reactions` (`ReactionConnection!`)	A list of Reactions left on the Issue. **Arguments for `reactions`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `content` (`ReactionContent`) Allows filtering Reactions by emoji. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`ReactionOrder`) Allows specifying the order in which reactions are returned.
`repository` (`Repository!`)	The repository associated with this node.
`resourcePath` (`URI!`)	The HTTP path permalink for this commit comment.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL permalink for this commit comment.
`userContentEdits` (`UserContentEditConnection`)	A list of edits to this content. **Arguments for `userContentEdits`**

	`after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`viewerCanDelete` (`Boolean!`)	Check if the current viewer can delete this object.
`viewerCanMinimize` (`Boolean!`)	Check if the current viewer can minimize this object.
`viewerCanReact` (`Boolean!`)	Can user react to this subject.
`viewerCanUpdate` (`Boolean!`)	Check if the current viewer can update this object.
`viewerCannotUpdateReasons` (`[CommentCannotUpdateReason!]!`)	Reasons why the current viewer can not update this comment.
`viewerDidAuthor` (`Boolean!`)	Did the viewer author this comment.

CommitCommentConnection

The connection type for CommitComment.

Fields for `CommitCommentConnection`

Name	Description
`edges` (`[CommitCommentEdge]`)	A list of edges.
`nodes` (`[CommitComment]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

CommitCommentEdge

An edge in a connection.

Fields for `CommitCommentEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.

`node` (`CommitComment`)	The item at the end of the edge.

CommitCommentThread

A thread of comments on a commit.

`CommitCommentThread` Implements

- `Node`
- `RepositoryNode`

Fields for `CommitCommentThread`

Name	Description
`comments` (`CommitCommentConnection!`)	The comments that exist in this thread. **Arguments for `comments`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`commit` (`Commit`)	The commit the comments were made on.
`path` (`String`)	The file the comments were made on.
`position` (`Int`)	The position in the diff for the commit that the comment was made on.
`repository` (`Repository!`)	The repository associated with this node.

CommitConnection

The connection type for Commit.

Fields for `CommitConnection`

Name	Description
`edges` (`[CommitEdge]`)	A list of edges.
`nodes` (`[Commit]`)	A list of nodes.

`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

CommitContributionsByRepository

This aggregates commits made by a user within one repository.

Fields for `CommitContributionsByRepository`

Name	Description
`contributions` (`CreatedCommitContributionConnection!`)	The commit contributions, each representing a day. **Arguments for `contributions`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`CommitContributionOrder`) Ordering options for commit contributions returned from the connection.
`repository` (`Repository!`)	The repository in which the commits were made.
`resourcePath` (`URI!`)	The HTTP path for the user's commits to the repository in this time range.
`url` (`URI!`)	The HTTP URL for the user's commits to the repository in this time range.

CommitEdge

An edge in a connection.

Fields for `CommitEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Commit`)	The item at the end of the edge.

CommitHistoryConnection

The connection type for Commit.

Fields for `CommitHistoryConnection`

Name	Description
`edges` (`[CommitEdge]`)	A list of edges.
`nodes` (`[Commit]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

CommitMessagePatternParameters

Parameters to be used for the commit_message_pattern rule.

Fields for `CommitMessagePatternParameters`

Name	Description
`name` (`String`)	How this rule will appear to users.
`negate` (`Boolean!`)	If true, the rule will fail if the pattern matches.
`operator` (`String!`)	The operator to use for matching.
`pattern` (`String!`)	The pattern to match with.

CommitterEmailPatternParameters

Parameters to be used for the committer_email_pattern rule.

Fields for `CommitterEmailPatternParameters`

Name	Description
`name` (`String`)	How this rule will appear to users.
`negate` (`Boolean!`)	If true, the rule will fail if the pattern matches.
`operator` (`String!`)	The operator to use for matching.
`pattern` (`String!`)	The pattern to match with.

Comparison

Represents a comparison between two commit revisions.

`Comparison` Implements

- `Node`

Fields for `Comparison`

Name	Description
`aheadBy` (`Int!`)	The number of commits ahead of the base branch.
`baseTarget` (`GitObject!`)	The base revision of this comparison.
`behindBy` (`Int!`)	The number of commits behind the base branch.
`commits` (`ComparisonCommitConnection!`)	The commits which compose this comparison. **Arguments for `commits`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`headTarget` (`GitObject!`)	The head revision of this comparison.
`status` (`ComparisonStatus!`)	The status of this comparison.

ComparisonCommitConnection

The connection type for Commit.

Fields for `ComparisonCommitConnection`

Name	Description
`authorCount` (`Int!`)	The total count of authors and co-authors across all commits.
`edges` (`[CommitEdge]`)	A list of edges.
`nodes` (`[Commit]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.

`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ConnectedEvent

Represents a `connected` event on a given issue or pull request.

`ConnectedEvent` Implements

- `Node`

Fields for `ConnectedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`isCrossRepository` (`Boolean!`)	Reference originated in a different repository.
`source` (`ReferencedSubject!`)	Issue or pull request that made the reference.
`subject` (`ReferencedSubject!`)	Issue or pull request which was connected.

ContributingGuidelines

The Contributing Guidelines for a repository.

Fields for `ContributingGuidelines`

Name	Description
`body` (`String`)	The body of the Contributing Guidelines.
`resourcePath` (`URI`)	The HTTP path for the Contributing Guidelines.
`url` (`URI`)	The HTTP URL for the Contributing Guidelines.

ContributionCalendar

A calendar of contributions made on GitHub by a user.

Fields for `ContributionCalendar`

Name	Description
`colors` (`[String!]!`)	A list of hex color codes used in this calendar. The darker the color, the more contributions it represents.
`isHalloween` (`Boolean!`)	Determine if the color set was chosen because

	it's currently Halloween.
`months` (`[ContributionCalendarMonth!]!`)	A list of the months of contributions in this calendar.
`totalContributions` (`Int!`)	The count of total contributions in the calendar.
`weeks` (`[ContributionCalendarWeek!]!`)	A list of the weeks of contributions in this calendar.

ContributionCalendarDay

Represents a single day of contributions on GitHub by a user.

Fields for `ContributionCalendarDay`

Name	Description
`color` (`String!`)	The hex color code that represents how many contributions were made on this day compared to others in the calendar.
`contributionCount` (`Int!`)	How many contributions were made by the user on this day.
`contributionLevel` (`ContributionLevel!`)	Indication of contributions, relative to other days. Can be used to indicate which color to represent this day on a calendar.
`date` (`Date!`)	The day this square represents.
`weekday` (`Int!`)	A number representing which day of the week this square represents, e.g., 1 is Monday.

ContributionCalendarMonth

A month of contributions in a user's contribution graph.

Fields for `ContributionCalendarMonth`

Name	Description
`firstDay` (`Date!`)	The date of the first day of this month.
`name` (`String!`)	The name of the month.
`totalWeeks` (`Int!`)	How many weeks started in this month.
`year` (`Int!`)	The year the month occurred in.

ContributionCalendarWeek

A week of contributions in a user's contribution graph.

Fields for `ContributionCalendarWeek`

Name	Description
`contributionDays` (`[ContributionCalendarDay!]!`)	The days of contributions in this week.
`firstDay` (`Date!`)	The date of the earliest square in this week.

ContributionsCollection

A contributions collection aggregates contributions such as opened issues and commits created by a user.

Fields for `ContributionsCollection`

Name	Description
`commitContributionsByRepository` (`[CommitContributionsByRepository!]!`)	Commit contributions made by the user, grouped by repository. **Arguments for `commitContributionsByRepository`** `maxRepositories` (`Int`) How many repositories should be included. The default value is `25`.
`contributionCalendar` (`ContributionCalendar!`)	A calendar of this user's contributions on GitHub.
`contributionYears` (`[Int!]!`)	The years the user has been making contributions with the most recent year first.
`doesEndInCurrentMonth` (`Boolean!`)	Determine if this collection's time span ends in the current month.
`earliestRestrictedContributionDate` (`Date`)	The date of the first restricted contribution the user made in this time period. Can only be non-null when the user has enabled private contribution counts.
`endedAt` (`DateTime!`)	The ending date and time of this collection.
`firstIssueContribution` (`CreatedIssueOrRestrictedContribution`)	The first issue the user opened on GitHub. This will be null if that issue was opened outside the collection's time range and ignoreTimeRange is false. If the issue is not visible but the user has opted to show private contributions, a RestrictedContribution will be returned.
`firstPullRequestContribution` (`CreatedPullRequestOrRestrictedContribution`)	The first pull request the user opened on GitHub. This will be null if that pull request was opened outside the collection's time range and ignoreTimeRange is not true. If the pull request is not visible but the user has opted to show private contributions, a RestrictedContribution will be returned.

`firstRepositoryContribution` (`CreatedRepositoryOrRestrictedContribution`)	The first repository the user created on GitHub. This will be null if that first repository was created outside the collection's time range and ignoreTimeRange is false. If the repository is not visible, then a RestrictedContribution is returned.
`hasActivityInThePast` (`Boolean!`)	Does the user have any more activity in the timeline that occurred prior to the collection's time range?.
`hasAnyContributions` (`Boolean!`)	Determine if there are any contributions in this collection.
`hasAnyRestrictedContributions` (`Boolean!`)	Determine if the user made any contributions in this time frame whose details are not visible because they were made in a private repository. Can only be true if the user enabled private contribution counts.
`isSingleDay` (`Boolean!`)	Whether or not the collector's time span is all within the same day.
`issueContributions` (`CreatedIssueContributionConnection!`)	A list of issues the user opened. **Arguments for `issueContributions`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `excludeFirst` (`Boolean`) Should the user's first issue ever be excluded from the result. The default value is `false`. `excludePopular` (`Boolean`) Should the user's most commented issue be excluded from the result. The default value is `false`. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`ContributionOrder`) Ordering options for contributions returned from the connection.
`issueContributionsByRepository` (`[IssueContributionsByRepository!]!`)	Issue contributions made by the user, grouped by repository. **Arguments for `issueContributionsByRepository`** `excludeFirst` (`Boolean`)

`excludeFirst` (Boolean)
Should the user's first issue ever be excluded from the result.
The default value is `false`.

`excludePopular` (Boolean)
Should the user's most commented issue be excluded from the result.
The default value is `false`.

`maxRepositories` (Int)
How many repositories should be included.
The default value is `25`.

`joinedGitHubContribution` (JoinedGitHubContribution)	When the user signed up for GitHub. This will be null if that sign up date falls outside the collection's time range and ignoreTimeRange is false.
`latestRestrictedContributionDate` (Date)	The date of the most recent restricted contribution the user made in this time period. Can only be non-null when the user has enabled private contribution counts.
`mostRecentCollectionWithActivity` (ContributionsCollection)	When this collection's time range does not include any activity from the user, use this to get a different collection from an earlier time range that does have activity.
`mostRecentCollectionWithoutActivity` (ContributionsCollection)	Returns a different contributions collection from an earlier time range than this one that does not have any contributions.
`popularIssueContribution` (CreatedIssueContribution)	The issue the user opened on GitHub that received the most comments in the specified time frame.
`popularPullRequestContribution` (CreatedPullRequestContribution)	The pull request the user opened on GitHub that received the most comments in the specified time frame.
`pullRequestContributions` (CreatedPullRequestContributionConnection!)	Pull request contributions made by the user.

Arguments for `pullRequestContributions`

`after` (String)
Returns the elements in the list that come after the specified cursor.

`before` (String)
Returns the elements in the list that come before the specified cursor.

`excludeFirst` (Boolean)
Should the user's first pull request ever be excluded from the result.
The default value is `false`.

`excludePopular` (Boolean)
Should the user's most commented pull request be excluded from the result.
The default value is `false`.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ContributionOrder`)
Ordering options for contributions returned from the connection.

`pullRequestContributionsByRepository` (`[PullRequestContributionsByRepository!]!`)

Pull request contributions made by the user, grouped by repository.

Arguments for `pullRequestContributionsByRepository`

`excludeFirst` (`Boolean`)
Should the user's first pull request ever be excluded from the result.
The default value is `false`.

`excludePopular` (`Boolean`)
Should the user's most commented pull request be excluded from the result.
The default value is `false`.

`maxRepositories` (`Int`)
How many repositories should be included.
The default value is `25`.

`pullRequestReviewContributions` (`CreatedPullRequestReviewContributionConnection!`)

Pull request review contributions made by the user. Returns the most recently submitted review for each PR reviewed by the user.

Arguments for `pullRequestReviewContributions`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)

Returns the last *n* elements from the list.

`orderBy` (`ContributionOrder`)
Ordering options for contributions returned from the connection.

`pullRequestReviewContributionsByRepository`

Pull request review contributions made by the

([PullRequestReviewContributionsByRepository!]!)	user, grouped by repository. **Arguments for `pullRequestReviewContributionsByRepository`** `maxRepositories` (Int) How many repositories should be included. The default value is `25`.
`repositoryContributions` (CreatedRepositoryContributionConnection!)	A list of repositories owned by the user that the user created in this time range. **Arguments for `repositoryContributions`** `after` (String) Returns the elements in the list that come after the specified cursor. `before` (String) Returns the elements in the list that come before the specified cursor. `excludeFirst` (Boolean) Should the user's first repository ever be excluded from the result. The default value is `false`. `first` (Int) Returns the first *n* elements from the list. `last` (Int) Returns the last *n* elements from the list. `orderBy` (ContributionOrder) Ordering options for contributions returned from the connection.
`restrictedContributionsCount` (Int!)	A count of contributions made by the user that the viewer cannot access. Only non-zero when the user has chosen to share their private contribution counts.
`startedAt` (DateTime!)	The beginning date and time of this collection.
`totalCommitContributions` (Int!)	How many commits were made by the user in this time span.
`totalIssueContributions` (Int!)	How many issues the user opened. **Arguments for `totalIssueContributions`** `excludeFirst` (Boolean) Should the user's first issue ever be excluded from this count. The default value is `false`. `excludePopular` (Boolean) Should the user's most commented issue

be excluded from this count.
The default value is `false`.

`totalPullRequestContributions` (`Int!`)

How many pull requests the user opened.

Arguments for `totalPullRequestContributions`

`excludeFirst` (`Boolean`)
Should the user's first pull request ever be excluded from this count.
The default value is `false`.

`excludePopular` (`Boolean`)
Should the user's most commented pull request be excluded from this count.
The default value is `false`.

`totalPullRequestReviewContributions` (`Int!`)

How many pull request reviews the user left.

`totalRepositoriesWithContributedCommits` (`Int!`)

How many different repositories the user committed to.

`totalRepositoriesWithContributedIssues` (`Int!`)

How many different repositories the user opened issues in.

Arguments for `totalRepositoriesWithContributedIssues`

`excludeFirst` (`Boolean`)
Should the user's first issue ever be excluded from this count.
The default value is `false`.

`excludePopular` (`Boolean`)
Should the user's most commented issue be excluded from this count.
The default value is `false`.

`totalRepositoriesWithContributedPullRequestReviews` (`Int!`)

How many different repositories the user left pull request reviews in.

`totalRepositoriesWithContributedPullRequests` (`Int!`)

How many different repositories the user opened pull requests in.

Arguments for `totalRepositoriesWithContributedPullRequests`

`excludeFirst` (`Boolean`)
Should the user's first pull request ever be excluded from this count.
The default value is `false`.

`excludePopular` (`Boolean`)
Should the user's most commented pull request be excluded from this count.

	The default value is `false`.
`totalRepositoryContributions` (`Int!`)	How many repositories the user created. **Arguments for `totalRepositoryContributions`** `excludeFirst` (`Boolean`) Should the user's first repository ever be excluded from this count. The default value is `false`.
`user` (`User!`)	The user who made the contributions in this collection.

ConvertToDraftEvent

Represents a `convert_to_draft` event on a given pull request.

`ConvertToDraftEvent` Implements

- `Node`
- `UniformResourceLocatable`

Fields for `ConvertToDraftEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`pullRequest` (`PullRequest!`)	PullRequest referenced by event.
`resourcePath` (`URI!`)	The HTTP path for this convert to draft event.
`url` (`URI!`)	The HTTP URL for this convert to draft event.

ConvertedNoteToIssueEvent

Represents a `converted_note_to_issue` event on a given issue or pull request.

`ConvertedNoteToIssueEvent` Implements

- `Node`

Fields for `ConvertedNoteToIssueEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.

`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`project` (`Project`)	Project referenced by event. **Preview notice** `project` is available under the Project event details preview. During the preview period, the API may change without notice.
`projectCard` (`ProjectCard`)	Project card referenced by this project event. **Preview notice** `projectCard` is available under the Project event details preview. During the preview period, the API may change without notice.
`projectColumnName` (`String!`)	Column name referenced by this project event. **Preview notice** `projectColumnName` is available under the Project event details preview. During the preview period, the API may change without notice.

ConvertedToDiscussionEvent

Represents a `converted_to_discussion` event on a given issue.

`ConvertedToDiscussionEvent` Implements

- `Node`

Fields for `ConvertedToDiscussionEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`discussion` (`Discussion`)	The discussion that the issue was converted into.

CreatedCommitContribution

Represents the contribution a user made by committing to a repository.

`CreatedCommitContribution` Implements

- `Contribution`

Fields for `CreatedCommitContribution`

Name	Description
`commitCount` (`Int!`)	How many commits were made on this day to this repository by the user.
`isRestricted` (`Boolean!`)	Whether this contribution is associated with a record you do not have access to. For example, your own 'first issue' contribution may have been made on a repository you can no longer access.
`occurredAt` (`DateTime!`)	When this contribution was made.
`repository` (`Repository!`)	The repository the user made a commit in.
`resourcePath` (`URI!`)	The HTTP path for this contribution.
`url` (`URI!`)	The HTTP URL for this contribution.
`user` (`User!`)	The user who made this contribution.

CreatedCommitContributionConnection

The connection type for CreatedCommitContribution.

Fields for `CreatedCommitContributionConnection`

Name	Description
`edges` (`[CreatedCommitContributionEdge]`)	A list of edges.
`nodes` (`[CreatedCommitContribution]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of commits across days and repositories in the connection.

CreatedCommitContributionEdge

An edge in a connection.

Fields for `CreatedCommitContributionEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`CreatedCommitContribution`)	The item at the end of the edge.

CreatedIssueContribution

Represents the contribution a user made on GitHub by opening an issue.

`CreatedIssueContribution` Implements

- `Contribution`

Fields for `CreatedIssueContribution`

Name	Description
`isRestricted` (`Boolean!`)	Whether this contribution is associated with a record you do not have access to. For example, your own 'first issue' contribution may have been made on a repository you can no longer access.
`issue` (`Issue!`)	The issue that was opened.
`occurredAt` (`DateTime!`)	When this contribution was made.
`resourcePath` (`URI!`)	The HTTP path for this contribution.
`url` (`URI!`)	The HTTP URL for this contribution.
`user` (`User!`)	The user who made this contribution.

CreatedIssueContributionConnection

The connection type for CreatedIssueContribution.

Fields for `CreatedIssueContributionConnection`

Name	Description
`edges` (`[CreatedIssueContributionEdge]`)	A list of edges.
`nodes` (`[CreatedIssueContribution]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

CreatedIssueContributionEdge

An edge in a connection.

Fields for `CreatedIssueContributionEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.

`node` (`CreatedIssueContribution`)	The item at the end of the edge.

CreatedPullRequestContribution

Represents the contribution a user made on GitHub by opening a pull request.

`CreatedPullRequestContribution` Implements

- `Contribution`

Fields for `CreatedPullRequestContribution`

Name	Description
`isRestricted` (`Boolean!`)	Whether this contribution is associated with a record you do not have access to. For example, your own 'first issue' contribution may have been made on a repository you can no longer access.
`occurredAt` (`DateTime!`)	When this contribution was made.
`pullRequest` (`PullRequest!`)	The pull request that was opened.
`resourcePath` (`URI!`)	The HTTP path for this contribution.
`url` (`URI!`)	The HTTP URL for this contribution.
`user` (`User!`)	The user who made this contribution.

CreatedPullRequestContributionConnection

The connection type for CreatedPullRequestContribution.

Fields for `CreatedPullRequestContributionConnection`

Name	Description
`edges` (`[CreatedPullRequestContributionEdge]`)	A list of edges.
`nodes` (`[CreatedPullRequestContribution]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

CreatedPullRequestContributionEdge

An edge in a connection.

Fields for `CreatedPullRequestContributionEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`CreatedPullRequestContribution`)	The item at the end of the edge.

CreatedPullRequestReviewContribution

Represents the contribution a user made by leaving a review on a pull request.

`CreatedPullRequestReviewContribution` Implements

- `Contribution`

Fields for `CreatedPullRequestReviewContribution`

Name	Description
`isRestricted` (`Boolean!`)	Whether this contribution is associated with a record you do not have access to. For example, your own 'first issue' contribution may have been made on a repository you can no longer access.
`occurredAt` (`DateTime!`)	When this contribution was made.
`pullRequest` (`PullRequest!`)	The pull request the user reviewed.
`pullRequestReview` (`PullRequestReview!`)	The review the user left on the pull request.
`repository` (`Repository!`)	The repository containing the pull request that the user reviewed.
`resourcePath` (`URI!`)	The HTTP path for this contribution.
`url` (`URI!`)	The HTTP URL for this contribution.
`user` (`User!`)	The user who made this contribution.

CreatedPullRequestReviewContributionConnection

The connection type for CreatedPullRequestReviewContribution.

Fields for `CreatedPullRequestReviewContributionConnection`

Name	Description
`edges` (`[CreatedPullRequestReviewContributionEdge]`)	A list of edges.
`nodes` (`[CreatedPullRequestReviewContribution]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

CreatedPullRequestReviewContributionEdge

An edge in a connection.

Fields for `CreatedPullRequestReviewContributionEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`CreatedPullRequestReviewContribution`)	The item at the end of the edge.

CreatedRepositoryContribution

Represents the contribution a user made on GitHub by creating a repository.

`CreatedRepositoryContribution` Implements

- `Contribution`

Fields for `CreatedRepositoryContribution`

Name	Description
`isRestricted` (`Boolean!`)	Whether this contribution is associated with a record you do not have access to. For example, your own 'first issue' contribution may have been made on a repository you can no longer access.
`occurredAt` (`DateTime!`)	When this contribution was made.
`repository` (`Repository!`)	The repository that was created.
`resourcePath` (`URI!`)	The HTTP path for this contribution.
`url` (`URI!`)	The HTTP URL for this contribution.
`user` (`User!`)	The user who made this contribution.

CreatedRepositoryContributionConnection

The connection type for CreatedRepositoryContribution.

Fields for `CreatedRepositoryContributionConnection`

Name	Description
`edges` (`[CreatedRepositoryContributionEdge]`)	A list of edges.
`nodes` (`[CreatedRepositoryContribution]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.

`totalCount` (`Int!`)	Identifies the total count of items in the connection.

CreatedRepositoryContributionEdge

An edge in a connection.

Fields for `CreatedRepositoryContributionEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`CreatedRepositoryContribution`)	The item at the end of the edge.

CrossReferencedEvent

Represents a mention made by one issue or pull request to another.

`CrossReferencedEvent` Implements

- `Node`
- `UniformResourceLocatable`

Fields for `CrossReferencedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`isCrossRepository` (`Boolean!`)	Reference originated in a different repository.
`referencedAt` (`DateTime!`)	Identifies when the reference was made.
`resourcePath` (`URI!`)	The HTTP path for this pull request.
`source` (`ReferencedSubject!`)	Issue or pull request that made the reference.
`target` (`ReferencedSubject!`)	Issue or pull request to which the reference was made.
`url` (`URI!`)	The HTTP URL for this pull request.
`willCloseTarget` (`Boolean!`)	Checks if the target will be closed when the source is merged.

DemilestonedEvent

Represents a `demilestoned` event on a given issue or pull request.

`DemilestonedEvent` Implements

- `Node`

Fields for `DemilestonedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`milestoneTitle` (`String!`)	Identifies the milestone title associated with the `demilestoned` event.
`subject` (`MilestoneItem!`)	Object referenced by event.

DependabotUpdate

A Dependabot Update for a dependency in a repository.

`DependabotUpdate` Implements

- `RepositoryNode`

Fields for `DependabotUpdate`

Name	Description
`error` (`DependabotUpdateError`)	The error from a dependency update.
`pullRequest` (`PullRequest`)	The associated pull request.
`repository` (`Repository!`)	The repository associated with this node.

DependabotUpdateError

An error produced from a Dependabot Update.

Fields for `DependabotUpdateError`

Name	Description
`body` (`String!`)	The body of the error.
`errorType` (`String!`)	The error code.
`title` (`String!`)	The title of the error.

DependencyGraphDependency

A dependency manifest entry.

Preview notice

`DependencyGraphDependency` is available under the Access to a repository's dependency graph preview. During the preview period, the API may change without notice.

Fields for `DependencyGraphDependency`

Name	Description
`hasDependencies` (`Boolean!`)	Does the dependency itself have dependencies?.
`packageLabel` (`String!`)	The original name of the package, as it appears in the manifest. **Deprecation notice** `packageLabel` is deprecated. `packageLabel` will be removed. Use normalized `packageName` field instead. Removal on 2022-10-01 UTC.
`packageManager` (`String`)	The dependency package manager.
`packageName` (`String!`)	The name of the package in the canonical form used by the package manager.
`repository` (`Repository`)	The repository containing the package.
`requirements` (`String!`)	The dependency version requirements.

DependencyGraphDependencyConnection

The connection type for DependencyGraphDependency.

Preview notice

`DependencyGraphDependencyConnection` is available under the Access to a repository's dependency graph preview. During the preview period, the API may change without notice.

Fields for `DependencyGraphDependencyConnection`

Name	Description
`edges` (`[DependencyGraphDependencyEdge]`)	A list of edges.
`nodes` (`[DependencyGraphDependency]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

DependencyGraphDependencyEdge

An edge in a connection.

> **Preview notice**
>
> `DependencyGraphDependencyEdge` is available under the Access to a repository's dependency graph preview. During the preview period, the API may change without notice.

Fields for `DependencyGraphDependencyEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`DependencyGraphDependency`)	The item at the end of the edge.

DependencyGraphManifest

Dependency manifest for a repository.

> **Preview notice**
>
> `DependencyGraphManifest` is available under the Access to a repository's dependency graph preview. During the preview period, the API may change without notice.

`DependencyGraphManifest` Implements

- `Node`

Fields for `DependencyGraphManifest`

Name	Description
`blobPath` (`String!`)	Path to view the manifest file blob.
`dependencies` (`DependencyGraphDependencyConnection`)	A list of manifest dependencies. **Arguments for `dependencies`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.

`dependenciesCount` (`Int`)	The number of dependencies listed in the manifest.
`exceedsMaxSize` (`Boolean!`)	Is the manifest too big to parse?.
`filename` (`String!`)	Fully qualified manifest filename.
`parseable` (`Boolean!`)	Were we able to parse the manifest?.
`repository` (`Repository!`)	The repository containing the manifest.

DependencyGraphManifestConnection

The connection type for DependencyGraphManifest.

> **Preview notice**
>
> `DependencyGraphManifestConnection` is available under the Access to a repository's dependency graph preview. During the preview period, the API may change without notice.

Fields for `DependencyGraphManifestConnection`

Name	Description
`edges` (`[DependencyGraphManifestEdge]`)	A list of edges.
`nodes` (`[DependencyGraphManifest]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

DependencyGraphManifestEdge

An edge in a connection.

> **Preview notice**
>
> `DependencyGraphManifestEdge` is available under the Access to a repository's dependency graph preview. During the preview period, the API may change without notice.

Fields for `DependencyGraphManifestEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`DependencyGraphManifest`)	The item at the end of the edge.

DeployKey

A repository deploy key.

`DeployKey` Implements

- `Node`

Fields for `DeployKey`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`key` (`String!`)	The deploy key.
`readOnly` (`Boolean!`)	Whether or not the deploy key is read only.
`title` (`String!`)	The deploy key title.
`verified` (`Boolean!`)	Whether or not the deploy key has been verified.

DeployKeyConnection

The connection type for DeployKey.

Fields for `DeployKeyConnection`

Name	Description
`edges` (`[DeployKeyEdge]`)	A list of edges.
`nodes` (`[DeployKey]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

DeployKeyEdge

An edge in a connection.

Fields for `DeployKeyEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`DeployKey`)	The item at the end of the edge.

DeployedEvent

Represents a `deployed` event on a given pull request.

`DeployedEvent` Implements

- `Node`

Fields for `DeployedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`deployment` (`Deployment!`)	The deployment associated with the `deployed` event.
`pullRequest` (`PullRequest!`)	PullRequest referenced by event.
`ref` (`Ref`)	The ref associated with the `deployed` event.

Deployment

Represents triggered deployment instance.

`Deployment` Implements

- `Node`

Fields for `Deployment`

Name	Description
`commit` (`Commit`)	Identifies the commit sha of the deployment.
`commitOid` (`String!`)	Identifies the oid of the deployment commit, even if the commit has been deleted.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`creator` (`Actor!`)	Identifies the actor who triggered the deployment.
`databaseId` (`Int`)	Identifies the primary key from the database.
`description` (`String`)	The deployment description.
`environment` (`String`)	The latest environment to which this deployment was made.
`latestEnvironment` (`String`)	The latest environment to which this deployment was made.
`latestStatus` (`DeploymentStatus`)	The latest status of this deployment.
`originalEnvironment` (`String`)	The original environment to which this

	deployment was made.
`payload` (`String`)	Extra information that a deployment system might need.
`ref` (`Ref`)	Identifies the Ref of the deployment, if the deployment was created by ref.
`repository` (`Repository!`)	Identifies the repository associated with the deployment.
`state` (`DeploymentState`)	The current state of the deployment.
`statuses` (`DeploymentStatusConnection`)	A list of statuses associated with the deployment. **Arguments for `statuses`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`task` (`String`)	The deployment task.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

DeploymentConnection

The connection type for Deployment.

Fields for `DeploymentConnection`

Name	Description
`edges` (`[DeploymentEdge]`)	A list of edges.
`nodes` (`[Deployment]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

DeploymentEdge

An edge in a connection.

Fields for `DeploymentEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Deployment`)	The item at the end of the edge.

DeploymentEnvironmentChangedEvent

Represents a `deployment_environment_changed` event on a given pull request.

`DeploymentEnvironmentChangedEvent` Implements

- `Node`

Fields for `DeploymentEnvironmentChangedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`deploymentStatus` (`DeploymentStatus!`)	The deployment status that updated the deployment environment.
`pullRequest` (`PullRequest!`)	PullRequest referenced by event.

DeploymentProtectionRule

A protection rule.

Fields for `DeploymentProtectionRule`

Name	Description
`databaseId` (`Int`)	Identifies the primary key from the database.
`preventSelfReview` (`Boolean`)	Whether deployments to this environment can be approved by the user who created the deployment.
`reviewers` (`DeploymentReviewerConnection!`)	The teams or users that can review the deployment. **Arguments for `reviewers`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`)

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`timeout` (`Int!`)	The timeout in minutes for this protection rule.
`type` (`DeploymentProtectionRuleType!`)	The type of protection rule.

DeploymentProtectionRuleConnection

The connection type for DeploymentProtectionRule.

Fields for `DeploymentProtectionRuleConnection`

Name	Description
`edges` (`[DeploymentProtectionRuleEdge]`)	A list of edges.
`nodes` (`[DeploymentProtectionRule]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

DeploymentProtectionRuleEdge

An edge in a connection.

Fields for `DeploymentProtectionRuleEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`DeploymentProtectionRule`)	The item at the end of the edge.

DeploymentRequest

A request to deploy a workflow run to an environment.

Fields for `DeploymentRequest`

Name	Description
`currentUserCanApprove` (`Boolean!`)	Whether or not the current user can approve the deployment.

`environment` (`Environment!`)	The target environment of the deployment.
`reviewers` (`DeploymentReviewerConnection!`)	The teams or users that can review the deployment. **Arguments for `reviewers`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`waitTimer` (`Int!`)	The wait timer in minutes configured in the environment.
`waitTimerStartedAt` (`DateTime`)	The wait timer in minutes configured in the environment.

DeploymentRequestConnection

The connection type for DeploymentRequest.

Fields for `DeploymentRequestConnection`

Name	Description
`edges` (`[DeploymentRequestEdge]`)	A list of edges.
`nodes` (`[DeploymentRequest]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

DeploymentRequestEdge

An edge in a connection.

Fields for `DeploymentRequestEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`DeploymentRequest`)	The item at the end of the edge.

`node` (`DeploymentRequest`)	The item at the end of the edge.

DeploymentReview

A deployment review.

`DeploymentReview` Implements

- `Node`

Fields for `DeploymentReview`

Name	Description
`comment` (`String!`)	The comment the user left.
`databaseId` (`Int`)	Identifies the primary key from the database.
`environments` (`EnvironmentConnection!`)	The environments approved or rejected. **Arguments for `environments`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`state` (`DeploymentReviewState!`)	The decision of the user.
`user` (`User!`)	The user that reviewed the deployment.

DeploymentReviewConnection

The connection type for DeploymentReview.

Fields for `DeploymentReviewConnection`

Name	Description
`edges` (`[DeploymentReviewEdge]`)	A list of edges.
`nodes` (`[DeploymentReview]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the

connection.

DeploymentReviewEdge

An edge in a connection.

Fields for `DeploymentReviewEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`DeploymentReview`)	The item at the end of the edge.

DeploymentReviewerConnection

The connection type for DeploymentReviewer.

Fields for `DeploymentReviewerConnection`

Name	Description
`edges` (`[DeploymentReviewerEdge]`)	A list of edges.
`nodes` (`[DeploymentReviewer]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

DeploymentReviewerEdge

An edge in a connection.

Fields for `DeploymentReviewerEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`DeploymentReviewer`)	The item at the end of the edge.

DeploymentStatus

Describes the status of a given deployment attempt.

`DeploymentStatus` Implements

- `Node`

Fields for `DeploymentStatus`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`creator` (`Actor!`)	Identifies the actor who triggered the deployment.
`deployment` (`Deployment!`)	Identifies the deployment associated with status.
`description` (`String`)	Identifies the description of the deployment.
`environment` (`String`)	Identifies the environment of the deployment at the time of this deployment status. **Preview notice** `environment` is available under the Deployments preview. During the preview period, the API may change without notice.
`environmentUrl` (`URI`)	Identifies the environment URL of the deployment.
`logUrl` (`URI`)	Identifies the log URL of the deployment.
`state` (`DeploymentStatusState!`)	Identifies the current state of the deployment.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

DeploymentStatusConnection

The connection type for DeploymentStatus.

Fields for `DeploymentStatusConnection`

Name	Description
`edges` (`[DeploymentStatusEdge]`)	A list of edges.
`nodes` (`[DeploymentStatus]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

DeploymentStatusEdge

An edge in a connection.

Fields for `DeploymentStatusEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`DeploymentStatus`)	The item at the end of the edge.

DisconnectedEvent

Represents a `disconnected` event on a given issue or pull request.

`DisconnectedEvent` Implements

- `Node`

Fields for `DisconnectedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`isCrossRepository` (`Boolean!`)	Reference originated in a different repository.
`source` (`ReferencedSubject!`)	Issue or pull request from which the issue was disconnected.
`subject` (`ReferencedSubject!`)	Issue or pull request which was disconnected.

Discussion

A discussion in a repository.

`Discussion` Implements

- `Closable`
- `Comment`
- `Deletable`
- `Labelable`
- `Lockable`
- `Node`
- `Reactable`
- `RepositoryNode`
- `Subscribable`
- `Updatable`
- `Votable`

Fields for `Discussion`

Name	Description

`activeLockReason` (`LockReason`)	Reason that the conversation was locked.
`answer` (`DiscussionComment`)	The comment chosen as this discussion's answer, if any.
`answerChosenAt` (`DateTime`)	The time when a user chose this discussion's answer, if answered.
`answerChosenBy` (`Actor`)	The user who chose this discussion's answer, if answered.
`author` (`Actor`)	The actor who authored the comment.
`authorAssociation` (`CommentAuthorAssociation!`)	Author's association with the subject of the comment.
`body` (`String!`)	The main text of the discussion post.
`bodyHTML` (`HTML!`)	The body rendered to HTML.
`bodyText` (`String!`)	The body rendered to text.
`category` (`DiscussionCategory!`)	The category for this discussion.
`closed` (`Boolean!`)	Indicates if the object is closed (definition of closed may depend on type).
`closedAt` (`DateTime`)	Identifies the date and time when the object was closed.
`comments` (`DiscussionCommentConnection!`)	The replies to the discussion. **Arguments for `comments`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`createdViaEmail` (`Boolean!`)	Check if this comment was created via an email reply.
`databaseId` (`Int`)	Identifies the primary key from the database.
`editor` (`Actor`)	The actor who edited the comment.
`includesCreatedEdit` (`Boolean!`)	Check if this comment was edited and includes an edit with the creation data.

`isAnswered` (`Boolean`)	Only return answered/unanswered discussions.
`labels` (`LabelConnection`)	A list of labels associated with the object. **Arguments for `labels`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`LabelOrder`) Ordering options for labels returned from the connection.
`lastEditedAt` (`DateTime`)	The moment the editor made the last edit.
`locked` (`Boolean!`)	`true` if the object is locked.
`number` (`Int!`)	The number identifying this discussion within the repository.
`poll` (`DiscussionPoll`)	The poll associated with this discussion, if one exists.
`publishedAt` (`DateTime`)	Identifies when the comment was published at.
`reactionGroups` (`[ReactionGroup!]`)	A list of reactions grouped by content left on the subject.
`reactions` (`ReactionConnection!`)	A list of Reactions left on the Issue. **Arguments for `reactions`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `content` (`ReactionContent`) Allows filtering Reactions by emoji. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`ReactionOrder`)

`orderBy` (`ReactionOrder`)
Allows specifying the order in which reactions are returned.

`repository` (`Repository!`)	The repository associated with this node.
`resourcePath` (`URI!`)	The path for this discussion.
`stateReason` (`DiscussionStateReason`)	Identifies the reason for the discussion's state.
`title` (`String!`)	The title of this discussion.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`upvoteCount` (`Int!`)	Number of upvotes that this subject has received.
`url` (`URI!`)	The URL for this discussion.
`userContentEdits` (`UserContentEditConnection`)	A list of edits to this content.

Arguments for `userContentEdits`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`viewerCanClose` (`Boolean!`)	Indicates if the object can be closed by the viewer.
`viewerCanDelete` (`Boolean!`)	Check if the current viewer can delete this object.
`viewerCanReact` (`Boolean!`)	Can user react to this subject.
`viewerCanReopen` (`Boolean!`)	Indicates if the object can be reopened by the viewer.
`viewerCanSubscribe` (`Boolean!`)	Check if the viewer is able to change their subscription status for the repository.
`viewerCanUpdate` (`Boolean!`)	Check if the current viewer can update this object.
`viewerCanUpvote` (`Boolean!`)	Whether or not the current user can add or remove an upvote on this subject.

`viewerDidAuthor` (`Boolean!`)	Did the viewer author this comment.
`viewerHasUpvoted` (`Boolean!`)	Whether or not the current user has already upvoted this subject.
`viewerSubscription` (`SubscriptionState`)	Identifies if the viewer is watching, not watching, or ignoring the subscribable entity.

DiscussionCategory

A category for discussions in a repository.

`DiscussionCategory` Implements

- `Node`
- `RepositoryNode`

Fields for `DiscussionCategory`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`description` (`String`)	A description of this category.
`emoji` (`String!`)	An emoji representing this category.
`emojiHTML` (`HTML!`)	This category's emoji rendered as HTML.
`isAnswerable` (`Boolean!`)	Whether or not discussions in this category support choosing an answer with the markDiscussionCommentAsAnswer mutation.
`name` (`String!`)	The name of this category.
`repository` (`Repository!`)	The repository associated with this node.
`slug` (`String!`)	The slug of this category.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

DiscussionCategoryConnection

The connection type for DiscussionCategory.

Fields for `DiscussionCategoryConnection`

Name	Description
`edges` (`[DiscussionCategoryEdge]`)	A list of edges.
`nodes` (`[DiscussionCategory]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.

`totalCount` (`Int!`)	Identifies the total count of items in the connection.

DiscussionCategoryEdge

An edge in a connection.

Fields for `DiscussionCategoryEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`DiscussionCategory`)	The item at the end of the edge.

DiscussionComment

A comment on a discussion.

`DiscussionComment` Implements

- `Comment`
- `Deletable`
- `Minimizable`
- `Node`
- `Reactable`
- `Updatable`
- `UpdatableComment`
- `Votable`

Fields for `DiscussionComment`

Name	Description
`author` (`Actor`)	The actor who authored the comment.
`authorAssociation` (`CommentAuthorAssociation!`)	Author's association with the subject of the comment.
`body` (`String!`)	The body as Markdown.
`bodyHTML` (`HTML!`)	The body rendered to HTML.
`bodyText` (`String!`)	The body rendered to text.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`createdViaEmail` (`Boolean!`)	Check if this comment was created via an email reply.
`databaseId` (`Int`)	Identifies the primary key from the database.
`deletedAt` (`DateTime`)	The time when this replied-to comment was

	deleted.
`discussion` (`Discussion`)	The discussion this comment was created in.
`editor` (`Actor`)	The actor who edited the comment.
`includesCreatedEdit` (`Boolean!`)	Check if this comment was edited and includes an edit with the creation data.
`isAnswer` (`Boolean!`)	Has this comment been chosen as the answer of its discussion?.
`isMinimized` (`Boolean!`)	Returns whether or not a comment has been minimized.
`lastEditedAt` (`DateTime`)	The moment the editor made the last edit.
`minimizedReason` (`String`)	Returns why the comment was minimized. One of `abuse`, `off-topic`, `outdated`, `resolved`, `duplicate` and `spam`. Note that the case and formatting of these values differs from the inputs to the `MinimizeComment` mutation.
`publishedAt` (`DateTime`)	Identifies when the comment was published at.
`reactionGroups` (`[ReactionGroup!]`)	A list of reactions grouped by content left on the subject.
`reactions` (`ReactionConnection!`)	A list of Reactions left on the Issue. **Arguments for `reactions`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `content` (`ReactionContent`) Allows filtering Reactions by emoji. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`ReactionOrder`) Allows specifying the order in which reactions are returned.
`replies` (`DiscussionCommentConnection!`)	The threaded replies to this comment. **Arguments for `replies`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`)

Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`replyTo` (`DiscussionComment`)	The discussion comment this comment is a reply to.
`resourcePath` (`URI!`)	The path for this discussion comment.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`upvoteCount` (`Int!`)	Number of upvotes that this subject has received.
`url` (`URI!`)	The URL for this discussion comment.
`userContentEdits` (`UserContentEditConnection`)	A list of edits to this content.

Arguments for `userContentEdits`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`viewerCanDelete` (`Boolean!`)	Check if the current viewer can delete this object.
`viewerCanMarkAsAnswer` (`Boolean!`)	Can the current user mark this comment as an answer?.
`viewerCanMinimize` (`Boolean!`)	Check if the current viewer can minimize this object.
`viewerCanReact` (`Boolean!`)	Can user react to this subject.
`viewerCanUnmarkAsAnswer` (`Boolean!`)	Can the current user unmark this comment as an answer?.
`viewerCanUpdate` (`Boolean!`)	Check if the current viewer can update this object.
`viewerCanUpvote` (`Boolean!`)	Whether or not the current user can add or remove an upvote on this subject.

	remove an upvote on this subject.
`viewerCannotUpdateReasons` (`[CommentCannotUpdateReason!]!`)	Reasons why the current viewer can not update this comment.
`viewerDidAuthor` (`Boolean!`)	Did the viewer author this comment.
`viewerHasUpvoted` (`Boolean!`)	Whether or not the current user has already upvoted this subject.

DiscussionCommentConnection

The connection type for DiscussionComment.

Fields for `DiscussionCommentConnection`

Name	Description
`edges` (`[DiscussionCommentEdge]`)	A list of edges.
`nodes` (`[DiscussionComment]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

DiscussionCommentEdge

An edge in a connection.

Fields for `DiscussionCommentEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`DiscussionComment`)	The item at the end of the edge.

DiscussionConnection

The connection type for Discussion.

Fields for `DiscussionConnection`

Name	Description
`edges` (`[DiscussionEdge]`)	A list of edges.
`nodes` (`[Discussion]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

DiscussionEdge

An edge in a connection.

Fields for `DiscussionEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Discussion`)	The item at the end of the edge.

DiscussionPoll

A poll for a discussion.

`DiscussionPoll` Implements

- `Node`

Fields for `DiscussionPoll`

Name	Description
`discussion` (`Discussion`)	The discussion that this poll belongs to.
`options` (`DiscussionPollOptionConnection`)	The options for this poll. **Arguments for `options`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`DiscussionPollOptionOrder`) How to order the options for the discussion poll.
`question` (`String!`)	The question that is being asked by this poll.
`totalVoteCount` (`Int!`)	The total number of votes that have been cast for this poll.

`viewerCanVote` (`Boolean!`)	Indicates if the viewer has permission to vote in this poll.
`viewerHasVoted` (`Boolean!`)	Indicates if the viewer has voted for any option in this poll.

DiscussionPollOption

An option for a discussion poll.

`DiscussionPollOption` Implements

- `Node`

Fields for `DiscussionPollOption`

Name	Description
`option` (`String!`)	The text for this option.
`poll` (`DiscussionPoll`)	The discussion poll that this option belongs to.
`totalVoteCount` (`Int!`)	The total number of votes that have been cast for this option.
`viewerHasVoted` (`Boolean!`)	Indicates if the viewer has voted for this option in the poll.

DiscussionPollOptionConnection

The connection type for DiscussionPollOption.

Fields for `DiscussionPollOptionConnection`

Name	Description
`edges` (`[DiscussionPollOptionEdge]`)	A list of edges.
`nodes` (`[DiscussionPollOption]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

DiscussionPollOptionEdge

An edge in a connection.

Fields for `DiscussionPollOptionEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.

`node` (`DiscussionPollOption`)	The item at the end of the edge.

DraftIssue

A draft issue within a project.

`DraftIssue` Implements

- `Node`

Fields for `DraftIssue`

Name	Description
`assignees` (`UserConnection!`)	A list of users to assigned to this draft issue. **Arguments for `assignees`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`body` (`String!`)	The body of the draft issue.
`bodyHTML` (`HTML!`)	The body of the draft issue rendered to HTML.
`bodyText` (`String!`)	The body of the draft issue rendered to text.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`creator` (`Actor`)	The actor who created this draft issue.
`projectV2Items` (`ProjectV2ItemConnection!`)	List of items linked with the draft issue (currently draft issue can be linked to only one item). **Arguments for `projectV2Items`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor.

	`first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`projectsV2` (`ProjectV2Connection!`)	Projects that link to this draft issue (currently draft issue can be linked to only one project). **Arguments for `projectsV2`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`title` (`String!`)	The title of the draft issue.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

Enterprise

An account to manage multiple organizations with consolidated policy and billing.

`Enterprise` Implements

- `AnnouncementBanner`
- `Node`

Fields for `Enterprise`

Name	Description
`announcement` (`String`)	The text of the announcement.
`announcementExpiresAt` (`DateTime`)	The expiration date of the announcement, if any.
`announcementUserDismissible` (`Boolean`)	Whether the announcement can be dismissed by the user.
`avatarUrl` (`URI!`)	A URL pointing to the enterprise's public avatar. **Arguments for**

Arguments for `avatarUrl`

`size` (`Int`)
The size of the resulting square image.

`billingInfo` (`EnterpriseBillingInfo`)	Enterprise billing information visible to enterprise billing managers.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`description` (`String`)	The description of the enterprise.
`descriptionHTML` (`HTML!`)	The description of the enterprise as HTML.
`location` (`String`)	The location of the enterprise.
`members` (`EnterpriseMemberConnection!`)	A list of users who are members of this enterprise.

Arguments for `members`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`deployment` (`EnterpriseUserDeployment`)
Only return members within the selected GitHub Enterprise deployment.

`first` (`Int`)
Returns the first *n* elements from the list.

`hasTwoFactorEnabled` (`Boolean`)
Only return members with this two-factor authentication status. Does not include members who only have an account on a GitHub Enterprise Server instance.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`EnterpriseMemberOrder`)
Ordering options for members returned from the connection.

`organizationLogins` (`[String!]`)
Only return members within the organizations with these logins.

`query` (`String`)
The search string to look for.

`role`
(`EnterpriseUserAccountMembershipRole`)

	The role of the user in the enterprise organization or server.
`name` (`String!`)	The name of the enterprise.
`organizations` (`OrganizationConnection!`)	A list of organizations that belong to this enterprise. **Arguments for `organizations`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`OrganizationOrder`) Ordering options for organizations returned from the connection. `query` (`String`) The search string to look for. `viewerOrganizationRole` (`RoleInOrganization`) The viewer's role in an organization.
`ownerInfo` (`EnterpriseOwnerInfo`)	Enterprise information visible to enterprise owners or enterprise owners' personal access tokens (classic) with read:enterprise or admin:enterprise scope.
`resourcePath` (`URI!`)	The HTTP path for this enterprise.
`slug` (`String!`)	The URL-friendly identifier for the enterprise.
`url` (`URI!`)	The HTTP URL for this enterprise.
`viewerIsAdmin` (`Boolean!`)	Is the current viewer an admin of this enterprise?.
`websiteUrl` (`URI`)	The URL of the enterprise website.

EnterpriseAdministratorConnection

The connection type for User.

Fields for `EnterpriseAdministratorConnection`

Name	Description
`edges` (`[EnterpriseAdministratorEdge]`)	A list of edges.
`nodes` (`[User]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

EnterpriseAdministratorEdge

A User who is an administrator of an enterprise.

Fields for `EnterpriseAdministratorEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`User`)	The item at the end of the edge.
`role` (`EnterpriseAdministratorRole!`)	The role of the administrator.

EnterpriseAdministratorInvitation

An invitation for a user to become an owner or billing manager of an enterprise.

`EnterpriseAdministratorInvitation` Implements

- `Node`

Fields for `EnterpriseAdministratorInvitation`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`email` (`String`)	The email of the person who was invited to the enterprise.
`enterprise` (`Enterprise!`)	The enterprise the invitation is for.
`invitee` (`User`)	The user who was invited to the enterprise.
`inviter` (`User`)	The user who created the invitation.
`role` (`EnterpriseAdministratorRole!`)	The invitee's pending role in the enterprise (owner or billing_manager).

EnterpriseAdministratorInvitationConnection

The connection type for EnterpriseAdministratorInvitation.

Fields for `EnterpriseAdministratorInvitationConnection`

Name	Description
`edges` (`[EnterpriseAdministratorInvitationEdge]`)	A list of edges.
`nodes` (`[EnterpriseAdministratorInvitation]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

EnterpriseAdministratorInvitationEdge

An edge in a connection.

Fields for `EnterpriseAdministratorInvitationEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`EnterpriseAdministratorInvitation`)	The item at the end of the edge.

EnterpriseBillingInfo

Enterprise billing information visible to enterprise billing managers and owners.

Fields for `EnterpriseBillingInfo`

Name	Description
`allLicensableUsersCount` (`Int!`)	The number of licenseable users/emails across the enterprise.
`assetPacks` (`Int!`)	The number of data packs used by all organizations owned by the enterprise.
`bandwidthQuota` (`Float!`)	The bandwidth quota in GB for all organizations owned by the enterprise.
`bandwidthUsage` (`Float!`)	The bandwidth usage in GB for all organizations owned by the enterprise.
`bandwidthUsagePercentage` (`Int!`)	The bandwidth usage as a percentage of the bandwidth quota.
`storageQuota` (`Float!`)	The storage quota in GB for all organizations owned by the enterprise.
`storageUsage` (`Float!`)	The storage usage in GB for all organizations owned by the enterprise.

	owned by the enterprise.
`storageUsagePercentage` (`Int!`)	The storage usage as a percentage of the storage quota.
`totalAvailableLicenses` (`Int!`)	The number of available licenses across all owned organizations based on the unique number of billable users.
`totalLicenses` (`Int!`)	The total number of licenses allocated.

EnterpriseConnection

The connection type for Enterprise.

Fields for `EnterpriseConnection`

Name	Description
`edges` (`[EnterpriseEdge]`)	A list of edges.
`nodes` (`[Enterprise]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

EnterpriseEdge

An edge in a connection.

Fields for `EnterpriseEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Enterprise`)	The item at the end of the edge.

EnterpriseFailedInvitationConnection

The connection type for OrganizationInvitation.

Fields for `EnterpriseFailedInvitationConnection`

Name	Description
`edges` (`[EnterpriseFailedInvitationEdge]`)	A list of edges.
`nodes` (`[OrganizationInvitation]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the

	connection.
`totalUniqueUserCount` (`Int!`)	Identifies the total count of unique users in the connection.

EnterpriseFailedInvitationEdge

A failed invitation to be a member in an enterprise organization.

Fields for `EnterpriseFailedInvitationEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`OrganizationInvitation`)	The item at the end of the edge.

EnterpriseIdentityProvider

An identity provider configured to provision identities for an enterprise. Visible to enterprise owners or enterprise owners' personal access tokens (classic) with read:enterprise or admin:enterprise scope.

`EnterpriseIdentityProvider` Implements

- `Node`

Fields for `EnterpriseIdentityProvider`

Name	Description
`digestMethod` (`SamlDigestAlgorithm`)	The digest algorithm used to sign SAML requests for the identity provider.
`enterprise` (`Enterprise`)	The enterprise this identity provider belongs to.
`externalIdentities` (`ExternalIdentityConnection!`)	ExternalIdentities provisioned by this identity provider.

Arguments for `externalIdentities`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`login` (`String`)
Filter to external identities with the users

Filter to external identities with the users login.

`membersOnly` (`Boolean`)
Filter to external identities with valid org membership only.

`userName` (`String`)
Filter to external identities with the users userName/NameID attribute.

`idpCertificate` (`X509Certificate`)	The x509 certificate used by the identity provider to sign assertions and responses.
`issuer` (`String`)	The Issuer Entity ID for the SAML identity provider.
`recoveryCodes` (`[String!]`)	Recovery codes that can be used by admins to access the enterprise if the identity provider is unavailable.
`signatureMethod` (`SamlSignatureAlgorithm`)	The signature algorithm used to sign SAML requests for the identity provider.
`ssoUrl` (`URI`)	The URL endpoint for the identity provider's SAML SSO.

EnterpriseMemberConnection

The connection type for EnterpriseMember.

Fields for `EnterpriseMemberConnection`

Name	Description
`edges` (`[EnterpriseMemberEdge]`)	A list of edges.
`nodes` (`[EnterpriseMember]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

EnterpriseMemberEdge

A User who is a member of an enterprise through one or more organizations.

Fields for `EnterpriseMemberEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`EnterpriseMember`)	The item at the end of the edge.

EnterpriseOrganizationMembershipConnection

The connection type for Organization.

Fields for `EnterpriseOrganizationMembershipConnection`

Name	Description
`edges` (`[EnterpriseOrganizationMembershipEdge]`)	A list of edges.
`nodes` (`[Organization]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

EnterpriseOrganizationMembershipEdge

An enterprise organization that a user is a member of.

Fields for `EnterpriseOrganizationMembershipEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Organization`)	The item at the end of the edge.
`role` (`EnterpriseUserAccountMembershipRole!`)	The role of the user in the enterprise membership.

EnterpriseOutsideCollaboratorConnection

The connection type for User.

Fields for `EnterpriseOutsideCollaboratorConnection`

Name	Description
`edges` (`[EnterpriseOutsideCollaboratorEdge]`)	A list of edges.
`nodes` (`[User]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

EnterpriseOutsideCollaboratorEdge

A User who is an outside collaborator of an enterprise through one or more

organizations.

Fields for `EnterpriseOutsideCollaboratorEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`User`)	The item at the end of the edge.
`repositories` (`EnterpriseRepositoryInfoConnection!`)	The enterprise organization repositories this user is a member of. **Arguments for `repositories`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`RepositoryOrder`) Ordering options for repositories.

EnterpriseOwnerInfo

Enterprise information visible to enterprise owners or enterprise owners' personal access tokens (classic) with read:enterprise or admin:enterprise scope.

Fields for `EnterpriseOwnerInfo`

Name	Description
`admins` (`EnterpriseAdministratorConnection!`)	A list of all of the administrators for this enterprise. **Arguments for `admins`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list.

`hasTwoFactorEnabled` (`Boolean`)
Only return administrators with this two-factor authentication status.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`EnterpriseMemberOrder`)
Ordering options for administrators returned from the connection.

`organizationLogins` (`[String!]`)
Only return members within the organizations with these logins.

`query` (`String`)
The search string to look for.

`role` (`EnterpriseAdministratorRole`)
The role to filter by.

`affiliatedUsersWithTwoFactorDisabled` (`UserConnection!`)	A list of users in the enterprise who currently have two-factor authentication disabled.

Arguments for `affiliatedUsersWithTwoFactorDisabled`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`affiliatedUsersWithTwoFactorDisabledExist` (`Boolean!`)	Whether or not affiliated users with two-factor authentication disabled exist in the enterprise.
`allowPrivateRepositoryForkingSetting` (`EnterpriseEnabledDisabledSettingValue!`)	The setting value for whether private repository forking is enabled for repositories in organizations in this enterprise.
`allowPrivateRepositoryForkingSettingOrganizations` (`OrganizationConnection!`)	A list of enterprise organizations configured with the provided private repository forking setting value.

Arguments for `allowPrivateRepositoryForkingSettingOrganizations`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`OrganizationOrder`)
Ordering options for organizations with this setting.

`value` (`Boolean!`)
The setting value to find organizations for.

`allowPrivateRepositoryForkingSettingPolicyValue` (`EnterpriseAllowPrivateRepositoryForkingPolicyValue`)	The value for the allow private repository forking policy on the enterprise.
`defaultRepositoryPermissionSetting` (`EnterpriseDefaultRepositoryPermissionSettingValue!`)	The setting value for base repository permissions for organizations in this enterprise.
`defaultRepositoryPermissionSettingOrganizations` (`OrganizationConnection!`)	A list of enterprise organizations configured with the provided base repository permission.

Arguments for `defaultRepositoryPermissionSettingOrganizations`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`OrganizationOrder`)
Ordering options for organizations with this setting.

`value` (`DefaultRepositoryPermissionField!`)
The permission to find organizations for.

`domains` (`VerifiableDomainConnection!`)	A list of domains owned by the enterprise. Visible to enterprise owners or enterprise owners' personal access tokens (classic) with admin:enterprise scope.

Arguments for `domains`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`isApproved` (`Boolean`)
Filter whether or not the domain is approved.

`isVerified` (`Boolean`)
Filter whether or not the domain is verified.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`VerifiableDomainOrder`)
Ordering options for verifiable domains returned.

`enterpriseServerInstallations` (`EnterpriseServerInstallationConnection!`)

Enterprise Server installations owned by the enterprise.

Arguments for `enterpriseServerInstallations`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`connectedOnly` (`Boolean`)
Whether or not to only return installations discovered via GitHub Connect.
The default value is `false`.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`EnterpriseServerInstallationOrder`)
Ordering options for Enterprise Server installations returned.

`failedInvitations` (`EnterpriseFailedInvitationConnection!`)

A list of failed invitations in the enterprise.

Arguments for `failedInvitations`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`query` (`String`)
The search string to look for.

`ipAllowListEnabledSetting` (`IpAllowListEnabledSettingValue!`)	The setting value for whether the enterprise has an IP allow list enabled.
`ipAllowListEntries` (`IpAllowListEntryConnection!`)	The IP addresses that are allowed to access resources owned by the enterprise. Visible to enterprise owners or enterprise owners' personal access tokens (classic) with admin:enterprise scope.

Arguments for `ipAllowListEntries`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`IpAllowListEntryOrder`)
Ordering options for IP allow list entries returned.

`ipAllowListForInstalledAppsEnabledSetting` (`IpAllowListForInstalledAppsEnabledSettingValue!`)	The setting value for whether the enterprise has IP allow list configuration for installed GitHub Apps enabled.
`isUpdatingDefaultRepositoryPermission` (`Boolean!`)	Whether or not the base repository permission is currently being updated.
`isUpdatingTwoFactorRequirement` (`Boolean!`)	Whether the two-factor authentication requirement is currently being enforced.
`membersCanChangeRepositoryVisibilitySetting` (`EnterpriseEnabledDisabledSettingValue!`)	The setting value for whether organization members with admin permissions on a repository can change repository visibility.

`membersCanChangeRepositoryVisibilitySettingOrganizations` (`OrganizationConnection!`)	A list of enterprise organizations configured with the provided can change repository visibility setting value. **Arguments for `membersCanChangeRepositoryVisibilitySettingOrganizations`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`OrganizationOrder`) Ordering options for organizations with this setting. `value` (`Boolean!`) The setting value to find organizations for.
`membersCanCreateInternalRepositoriesSetting` (`Boolean`)	The setting value for whether members of organizations in the enterprise can create internal repositories.
`membersCanCreatePrivateRepositoriesSetting` (`Boolean`)	The setting value for whether members of organizations in the enterprise can create private repositories.
`membersCanCreatePublicRepositoriesSetting` (`Boolean`)	The setting value for whether members of organizations in the enterprise can create public repositories.
`membersCanCreateRepositoriesSetting` (`EnterpriseMembersCanCreateRepositoriesSettingValue`)	The setting value for whether members of organizations in the enterprise can create repositories.
`membersCanCreateRepositoriesSettingOrganizations` (`OrganizationConnection!`)	A list of enterprise organizations configured with the provided repository creation setting value. **Arguments for `membersCanCreateRepositoriesSettingOrganizations`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`)

Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`OrganizationOrder`)
Ordering options for organizations with this setting.

`value` (`OrganizationMembersCanCreateRepositoriesSettingValue!`)
The setting to find organizations for.

`membersCanDeleteIssuesSetting` (`EnterpriseEnabledDisabledSettingValue!`)	The setting value for whether members with admin permissions for repositories can delete issues.
`membersCanDeleteIssuesSettingOrganizations` (`OrganizationConnection!`)	A list of enterprise organizations configured with the provided members can delete issues setting value.

Arguments for `membersCanDeleteIssuesSettingOrganizations`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`OrganizationOrder`)
Ordering options for organizations with this setting.

`value` (`Boolean!`)
The setting value to find organizations for.

`membersCanDeleteRepositoriesSetting` (`EnterpriseEnabledDisabledSettingValue!`)	The setting value for whether members with admin permissions for repositories can delete or transfer repositories.
`membersCanDeleteRepositoriesSettingOrganizations` (`OrganizationConnection!`)	A list of enterprise organizations configured with the provided members can delete repositories setting value.

Arguments for `membersCanDeleteRepositoriesSettingOrganizations`

`after` (`String`)
Returns the elements in the list that come

Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`OrganizationOrder`)
Ordering options for organizations with this setting.

`value` (`Boolean!`)
The setting value to find organizations for.

`membersCanInviteCollaboratorsSetting` (`EnterpriseEnabledDisabledSettingValue!`)	The setting value for whether members of organizations in the enterprise can invite outside collaborators.
`membersCanInviteCollaboratorsSettingOrganizations` (`OrganizationConnection!`)	A list of enterprise organizations configured with the provided members can invite collaborators setting value.

Arguments for `membersCanInviteCollaboratorsSettingOrganizations`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`OrganizationOrder`)
Ordering options for organizations with this setting.

`value` (`Boolean!`)
The setting value to find organizations for.

`membersCanMakePurchasesSetting` (`EnterpriseMembersCanMakePurchasesSettingValue!`)	Indicates whether members of this enterprise's organizations can purchase additional services for those organizations.
`membersCanUpdateProtectedBranchesSetting` (`EnterpriseEnabledDisabledSettingValue!`)	The setting value for whether members with admin permissions for repositories can update protected branches.

`membersCanUpdateProtectedBranchesSettingOrganizations` (`OrganizationConnection!`)	A list of enterprise organizations configured with the provided members can update protected branches setting value. **Arguments for `membersCanUpdateProtectedBranchesSettingOrganizations`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`OrganizationOrder`) Ordering options for organizations with this setting. `value` (`Boolean!`) The setting value to find organizations for.
`membersCanViewDependencyInsightsSetting` (`EnterpriseEnabledDisabledSettingValue!`)	The setting value for whether members can view dependency insights.
`membersCanViewDependencyInsightsSettingOrganizations` (`OrganizationConnection!`)	A list of enterprise organizations configured with the provided members can view dependency insights setting value. **Arguments for `membersCanViewDependencyInsightsSettingOrganizations`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`OrganizationOrder`) Ordering options for organizations with this setting. `value` (`Boolean!`) The setting value to find organizations for.

`notificationDeliveryRestrictionEnabledSetting` (`NotificationRestrictionSettingValue!`)	Indicates if email notification delivery for this enterprise is restricted to verified or approved domains.
`oidcProvider` (`OIDCProvider`)	The OIDC Identity Provider for the enterprise.
`organizationProjectsSetting` (`EnterpriseEnabledDisabledSettingValue!`)	The setting value for whether organization projects are enabled for organizations in this enterprise.
`organizationProjectsSettingOrganizations` (`OrganizationConnection!`)	A list of enterprise organizations configured with the provided organization projects setting value. **Arguments for `organizationProjectsSettingOrganizations`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`OrganizationOrder`) Ordering options for organizations with this setting. `value` (`Boolean!`) The setting value to find organizations for.
`outsideCollaborators` (`EnterpriseOutsideCollaboratorConnection!`)	A list of outside collaborators across the repositories in the enterprise. **Arguments for `outsideCollaborators`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `hasTwoFactorEnabled` (`Boolean`) Only return outside collaborators with this two-factor authentication status. `last` (`Int`) Returns the last *n* elements from the list.

`login` (`String`)
The login of one specific outside collaborator.

`orderBy` (`EnterpriseMemberOrder`)
Ordering options for outside collaborators returned from the connection.

`organizationLogins` (`[String!]`)
Only return outside collaborators within the organizations with these logins.

`query` (`String`)
The search string to look for.

`visibility` (`RepositoryVisibility`)
Only return outside collaborators on repositories with this visibility.

`pendingAdminInvitations` (`EnterpriseAdministratorInvitationConnection!`)

A list of pending administrator invitations for the enterprise.

Arguments for `pendingAdminInvitations`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`EnterpriseAdministratorInvitationOrder`)
Ordering options for pending enterprise administrator invitations returned from the connection.

`query` (`String`)
The search string to look for.

`role` (`EnterpriseAdministratorRole`)
The role to filter by.

`pendingCollaboratorInvitations` (`RepositoryInvitationConnection!`)

A list of pending collaborator invitations across the repositories in the enterprise.

Arguments for `pendingCollaboratorInvitations`

`after` (`String`)
Returns the elements in the list that come

after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`RepositoryInvitationOrder`)
Ordering options for pending repository collaborator invitations returned from the connection.

`query` (`String`)
The search string to look for.

`pendingMemberInvitations` (`EnterprisePendingMemberInvitationConnection!`)

A list of pending member invitations for organizations in the enterprise.

Arguments for `pendingMemberInvitations`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`invitationSource` (`OrganizationInvitationSource`)
Only return invitations matching this invitation source.

`last` (`Int`)
Returns the last *n* elements from the list.

`organizationLogins` (`[String!]`)
Only return invitations within the organizations with these logins.

`query` (`String`)
The search string to look for.

`repositoryProjectsSetting` (`EnterpriseEnabledDisabledSettingValue!`)

The setting value for whether repository projects are enabled in this enterprise.

`repositoryProjectsSettingOrganizations` (`OrganizationConnection!`)

A list of enterprise organizations configured with the provided repository projects setting value.

Arguments for `repositoryProjectsSettingOrganizations`

`repositoryProjectsSettingOrganizations`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`OrganizationOrder`)
Ordering options for organizations with this setting.

`value` (`Boolean!`)
The setting value to find organizations for.

`samlIdentityProvider` (`EnterpriseIdentityProvider`)

The SAML Identity Provider for the enterprise.

`samlIdentityProviderSettingOrganizations` (`OrganizationConnection!`)

A list of enterprise organizations configured with the SAML single sign-on setting value.

Arguments for `samlIdentityProviderSettingOrganizations`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`OrganizationOrder`)
Ordering options for organizations with this setting.

`value` (`IdentityProviderConfigurationState!`)
The setting value to find organizations for.

`supportEntitlements` (`EnterpriseMemberConnection!`)

A list of members with a support entitlement.

Arguments for `supportEntitlements`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`EnterpriseMemberOrder`)
Ordering options for support entitlement users returned from the connection.

`teamDiscussionsSetting` (`EnterpriseEnabledDisabledSettingValue!`)	The setting value for whether team discussions are enabled for organizations in this enterprise.
`teamDiscussionsSettingOrganizations` (`OrganizationConnection!`)	A list of enterprise organizations configured with the provided team discussions setting value.

Arguments for `teamDiscussionsSettingOrganizations`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`OrganizationOrder`)
Ordering options for organizations with this setting.

`value` (`Boolean!`)
The setting value to find organizations for.

`twoFactorRequiredSetting` (`EnterpriseEnabledSettingValue!`)	The setting value for whether the enterprise requires two-factor authentication for its organizations and users.
`twoFactorRequiredSettingOrganizations` (`OrganizationConnection!`)	A list of enterprise organizations configured with the two-factor authentication setting value.

Arguments for `twoFactorRequiredSettingOrganizations`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`OrganizationOrder`)
Ordering options for organizations with this setting.

`value` (`Boolean!`)
The setting value to find organizations for.

EnterprisePendingMemberInvitationConnection

The connection type for OrganizationInvitation.

Fields for `EnterprisePendingMemberInvitationConnection`

Name	Description
`edges` (`[EnterprisePendingMemberInvitationEdge]`)	A list of edges.
`nodes` (`[OrganizationInvitation]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.
`totalUniqueUserCount` (`Int!`)	Identifies the total count of unique users in the connection.

EnterprisePendingMemberInvitationEdge

An invitation to be a member in an enterprise organization.

Fields for `EnterprisePendingMemberInvitationEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`OrganizationInvitation`)	The item at the end of the edge.

EnterpriseRepositoryInfo

A subset of repository information queryable from an enterprise.

`EnterpriseRepositoryInfo` Implements

- `Node`

Fields for `EnterpriseRepositoryInfo`

Name	Description
`isPrivate` (`Boolean!`)	Identifies if the repository is private or internal.
`name` (`String!`)	The repository's name.
`nameWithOwner` (`String!`)	The repository's name with owner.

EnterpriseRepositoryInfoConnection

The connection type for EnterpriseRepositoryInfo.

Fields for `EnterpriseRepositoryInfoConnection`

Name	Description
`edges` (`[EnterpriseRepositoryInfoEdge]`)	A list of edges.
`nodes` (`[EnterpriseRepositoryInfo]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

EnterpriseRepositoryInfoEdge

An edge in a connection.

Fields for `EnterpriseRepositoryInfoEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`EnterpriseRepositoryInfo`)	The item at the end of the edge.

EnterpriseServerInstallation

An Enterprise Server installation.

`EnterpriseServerInstallation` Implements

- `Node`

Fields for `EnterpriseServerInstallation`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`customerName` (`String!`)	The customer name to which the Enterprise Server installation belongs.
`hostName` (`String!`)	The host name of the Enterprise Server installation.
`isConnected` (`Boolean!`)	Whether or not the installation is connected to an Enterprise Server installation via GitHub Connect.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`userAccounts` (`EnterpriseServerUserAccountConnection!`)	User accounts on this Enterprise Server installation. **Arguments for `userAccounts`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`EnterpriseServerUserAccountOrder`) Ordering options for Enterprise Server user accounts returned from the connection.
`userAccountsUploads` (`EnterpriseServerUserAccountsUploadConnection!`)	User accounts uploads for the Enterprise Server installation. **Arguments for `userAccountsUploads`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.

`orderBy` (`EnterpriseServerUserAccountsUploadOrder`) Ordering options for Enterprise Server user accounts uploads returned from the connection.

EnterpriseServerInstallationConnection

The connection type for EnterpriseServerInstallation.

Fields for `EnterpriseServerInstallationConnection`

Name	Description
`edges` (`[EnterpriseServerInstallationEdge]`)	A list of edges.
`nodes` (`[EnterpriseServerInstallation]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

EnterpriseServerInstallationEdge

An edge in a connection.

Fields for `EnterpriseServerInstallationEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`EnterpriseServerInstallation`)	The item at the end of the edge.

EnterpriseServerInstallationMembershipConnection

The connection type for EnterpriseServerInstallation.

Fields for `EnterpriseServerInstallationMembershipConnection`

Name	Description
`edges` (`[EnterpriseServerInstallationMembershipEdge]`)	A list of edges.
`nodes` (`[EnterpriseServerInstallation]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the

`totalCount` (`Int!`)	Identifies the total count of items in the connection.

EnterpriseServerInstallationMembershipEdge

An Enterprise Server installation that a user is a member of.

Fields for `EnterpriseServerInstallationMembershipEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`EnterpriseServerInstallation`)	The item at the end of the edge.
`role` (`EnterpriseUserAccountMembershipRole!`)	The role of the user in the enterprise membership.

EnterpriseServerUserAccount

A user account on an Enterprise Server installation.

`EnterpriseServerUserAccount` Implements

- `Node`

Fields for `EnterpriseServerUserAccount`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`emails` (`EnterpriseServerUserAccountEmailConnection!`)	User emails belonging to this user account. **Arguments for `emails`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`EnterpriseServerUserAccountEmailOrder`) Ordering options for Enterprise Server user account emails returned from the connection.

`enterpriseServerInstallation` (`EnterpriseServerInstallation!`)	The Enterprise Server installation on which this user account exists.
`isSiteAdmin` (`Boolean!`)	Whether the user account is a site administrator on the Enterprise Server installation.
`login` (`String!`)	The login of the user account on the Enterprise Server installation.
`profileName` (`String`)	The profile name of the user account on the Enterprise Server installation.
`remoteCreatedAt` (`DateTime!`)	The date and time when the user account was created on the Enterprise Server installation.
`remoteUserId` (`Int!`)	The ID of the user account on the Enterprise Server installation.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

EnterpriseServerUserAccountConnection

The connection type for EnterpriseServerUserAccount.

Fields for `EnterpriseServerUserAccountConnection`

Name	Description
`edges` (`[EnterpriseServerUserAccountEdge]`)	A list of edges.
`nodes` (`[EnterpriseServerUserAccount]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

EnterpriseServerUserAccountEdge

An edge in a connection.

Fields for `EnterpriseServerUserAccountEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`EnterpriseServerUserAccount`)	The item at the end of the edge.

EnterpriseServerUserAccountEmail

An email belonging to a user account on an Enterprise Server installation.

`EnterpriseServerUserAccountEmail` Implements

- `Node`

Fields for `EnterpriseServerUserAccountEmail`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`email` (`String!`)	The email address.
`isPrimary` (`Boolean!`)	Indicates whether this is the primary email of the associated user account.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`userAccount` (`EnterpriseServerUserAccount!`)	The user account to which the email belongs.

EnterpriseServerUserAccountEmailConnection

The connection type for EnterpriseServerUserAccountEmail.

Fields for `EnterpriseServerUserAccountEmailConnection`

Name	Description
`edges` (`[EnterpriseServerUserAccountEmailEdge]`)	A list of edges.
`nodes` (`[EnterpriseServerUserAccountEmail]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

EnterpriseServerUserAccountEmailEdge

An edge in a connection.

Fields for `EnterpriseServerUserAccountEmailEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`EnterpriseServerUserAccountEmail`)	The item at the end of the edge.

EnterpriseServerUserAccountsUpload

A user accounts upload from an Enterprise Server installation.

`EnterpriseServerUserAccountsUpload` Implements

- `Node`

Fields for `EnterpriseServerUserAccountsUpload`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`enterprise` (`Enterprise!`)	The enterprise to which this upload belongs.
`enterpriseServerInstallation` (`EnterpriseServerInstallation!`)	The Enterprise Server installation for which this upload was generated.
`name` (`String!`)	The name of the file uploaded.
`syncState` (`EnterpriseServerUserAccountsUploadSyncState!`)	The synchronization state of the upload.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

EnterpriseServerUserAccountsUploadConnection

The connection type for EnterpriseServerUserAccountsUpload.

Fields for `EnterpriseServerUserAccountsUploadConnection`

Name	Description
`edges` (`[EnterpriseServerUserAccountsUploadEdge]`)	A list of edges.
`nodes` (`[EnterpriseServerUserAccountsUpload]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

EnterpriseServerUserAccountsUploadEdge

An edge in a connection.

Fields for `EnterpriseServerUserAccountsUploadEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`EnterpriseServerUserAccountsUpload`)	The item at the end of the edge.

EnterpriseUserAccount

An account for a user who is an admin of an enterprise or a member of an enterprise through one or more organizations.

`EnterpriseUserAccount` Implements

- `Actor`
- `Node`

Fields for `EnterpriseUserAccount`

Name	Description
`avatarUrl` (`URI!`)	A URL pointing to the enterprise user account's public avatar. **Arguments for `avatarUrl`** `size` (`Int`) The size of the resulting square image.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`enterprise` (`Enterprise!`)	The enterprise in which this user account exists.
`enterpriseInstallations` (`EnterpriseServerInstallationMembershipConnection!`)	A list of Enterprise Server installations this user is a member of. **Arguments for `enterpriseInstallations`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`EnterpriseServerInstallationOrder`) Ordering options for installations returned from the connection. `query` (`String`) The search string to look for. `role` (`EnterpriseUserAccountMembershipRole`) The role of the user in the installation.

`login` (`String!`)	An identifier for the enterprise user account, a login or email address.
`name` (`String`)	The name of the enterprise user account.
`organizations` (`EnterpriseOrganizationMembershipConnection!`)	A list of enterprise organizations this user is a member of. **Arguments for `organizations`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`OrganizationOrder`) Ordering options for organizations returned from the connection. `query` (`String`) The search string to look for. `role` (`EnterpriseUserAccountMembershipRole`) The role of the user in the enterprise organization.
`resourcePath` (`URI!`)	The HTTP path for this user.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this user.
`user` (`User`)	The user within the enterprise.

Environment

An environment.

`Environment` Implements

- `Node`

Fields for `Environment`

Name	Description

`databaseId` (`Int`)	Identifies the primary key from the database.
`name` (`String!`)	The name of the environment.
`protectionRules` (`DeploymentProtectionRuleConnection!`)	The protection rules defined for this environment. **Arguments for `protectionRules`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.

EnvironmentConnection

The connection type for Environment.

Fields for `EnvironmentConnection`

Name	Description
`edges` (`[EnvironmentEdge]`)	A list of edges.
`nodes` (`[Environment]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

EnvironmentEdge

An edge in a connection.

Fields for `EnvironmentEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Environment`)	The item at the end of the edge.

ExternalIdentity

An external identity provisioned by SAML SSO or SCIM. If SAML is configured on the organization, the external identity is visible to (1) organization owners, (2) organization owners' personal access tokens (classic) with read:org or admin:org scope, (3) GitHub App with an installation token with read or write access to members. If SAML is configured on the enterprise, the external identity is visible to (1) enterprise owners, (2) enterprise owners' personal access tokens (classic) with read:enterprise or admin:enterprise scope.

`ExternalIdentity` Implements

- `Node`

Fields for `ExternalIdentity`

Name	Description
`guid` (`String!`)	The GUID for this identity.
`organizationInvitation` (`OrganizationInvitation`)	Organization invitation for this SCIM-provisioned external identity.
`samlIdentity` (`ExternalIdentitySamlAttributes`)	SAML Identity attributes.
`scimIdentity` (`ExternalIdentityScimAttributes`)	SCIM Identity attributes.
`user` (`User`)	User linked to this external identity. Will be NULL if this identity has not been claimed by an organization member.

ExternalIdentityAttribute

An attribute for the External Identity attributes collection.

Fields for `ExternalIdentityAttribute`

Name	Description
`metadata` (`String`)	The attribute metadata as JSON.
`name` (`String!`)	The attribute name.
`value` (`String!`)	The attribute value.

ExternalIdentityConnection

The connection type for ExternalIdentity.

Fields for `ExternalIdentityConnection`

Name	Description
`edges` (`[ExternalIdentityEdge]`)	A list of edges.
`nodes` (`[ExternalIdentity]`)	A list of nodes.

`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ExternalIdentityEdge

An edge in a connection.

Fields for `ExternalIdentityEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`ExternalIdentity`)	The item at the end of the edge.

ExternalIdentitySamlAttributes

SAML attributes for the External Identity.

Fields for `ExternalIdentitySamlAttributes`

Name	Description
`attributes` (`[ExternalIdentityAttribute!]!`)	SAML Identity attributes.
`emails` (`[UserEmailMetadata!]`)	The emails associated with the SAML identity.
`familyName` (`String`)	Family name of the SAML identity.
`givenName` (`String`)	Given name of the SAML identity.
`groups` (`[String!]`)	The groups linked to this identity in IDP.
`nameId` (`String`)	The NameID of the SAML identity.
`username` (`String`)	The userName of the SAML identity.

ExternalIdentityScimAttributes

SCIM attributes for the External Identity.

Fields for `ExternalIdentityScimAttributes`

Name	Description
`emails` (`[UserEmailMetadata!]`)	The emails associated with the SCIM identity.
`familyName` (`String`)	Family name of the SCIM identity.
`givenName` (`String`)	Given name of the SCIM identity.
`groups` (`[String!]`)	The groups linked to this identity in IDP.

`groups` (`[String!]`)	The groups linked to this identity in IDP.
`username` (`String`)	The userName of the SCIM identity.

FollowerConnection

The connection type for User.

Fields for `FollowerConnection`

Name	Description
`edges` (`[UserEdge]`)	A list of edges.
`nodes` (`[User]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

FollowingConnection

The connection type for User.

Fields for `FollowingConnection`

Name	Description
`edges` (`[UserEdge]`)	A list of edges.
`nodes` (`[User]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

FundingLink

A funding platform link for a repository.

Fields for `FundingLink`

Name	Description
`platform` (`FundingPlatform!`)	The funding platform this link is for.
`url` (`URI!`)	The configured URL for this funding link.

GenericHovercardContext

A generic hovercard context with a message and icon.

`GenericHovercardContext` Implements

- `HovercardContext`

Fields for `GenericHovercardContext`

Name	Description
`message` (`String!`)	A string describing this context.
`octicon` (`String!`)	An octicon to accompany this context.

Gist

A Gist.

`Gist` Implements

- `Node`
- `Starrable`
- `UniformResourceLocatable`

Fields for `Gist`

Name	Description
`comments` (`GistCommentConnection!`)	A list of comments associated with the gist. **Arguments for `comments`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`description` (`String`)	The gist description.
`files` (`[GistFile]`)	The files in this gist. **Arguments for `files`** `limit` (`Int`) The maximum number of files to return. The default value is `10`

The default value is `10`.

`oid` (`GitObjectID`)
The oid of the files to return.

`forks` (`GistConnection!`)	A list of forks associated with the gist. **Arguments for `forks`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`GistOrder`) Ordering options for gists returned from the connection.
`isFork` (`Boolean!`)	Identifies if the gist is a fork.
`isPublic` (`Boolean!`)	Whether the gist is public or not.
`name` (`String!`)	The gist name.
`owner` (`RepositoryOwner`)	The gist owner.
`pushedAt` (`DateTime`)	Identifies when the gist was last pushed to.
`resourcePath` (`URI!`)	The HTML path to this resource.
`stargazerCount` (`Int!`)	Returns a count of how many stargazers there are on this object.
`stargazers` (`StargazerConnection!`)	A list of users who have starred this starrable. **Arguments for `stargazers`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.

	`orderBy` (`StarOrder`) Order for connection.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this Gist.
`viewerHasStarred` (`Boolean!`)	Returns a boolean indicating whether the viewing user has starred this starrable.

GistComment

Represents a comment on an Gist.

`GistComment` Implements

- `Comment`
- `Deletable`
- `Minimizable`
- `Node`
- `Updatable`
- `UpdatableComment`

Fields for `GistComment`

Name	Description
`author` (`Actor`)	The actor who authored the comment.
`authorAssociation` (`CommentAuthorAssociation!`)	Author's association with the gist.
`body` (`String!`)	Identifies the comment body.
`bodyHTML` (`HTML!`)	The body rendered to HTML.
`bodyText` (`String!`)	The body rendered to text.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`createdViaEmail` (`Boolean!`)	Check if this comment was created via an email reply.
`databaseId` (`Int`)	Identifies the primary key from the database.
`editor` (`Actor`)	The actor who edited the comment.
`gist` (`Gist!`)	The associated gist.
`includesCreatedEdit` (`Boolean!`)	Check if this comment was edited and includes an edit with the creation data.
`isMinimized` (`Boolean!`)	Returns whether or not a comment has been minimized.

`lastEditedAt` (`DateTime`)	The moment the editor made the last edit.
`minimizedReason` (`String`)	Returns why the comment was minimized. One of `abuse` , `off-topic` , `outdated` , `resolved` , `duplicate` and `spam` . Note that the case and formatting of these values differs from the inputs to the `MinimizeComment` mutation.
`publishedAt` (`DateTime`)	Identifies when the comment was published at.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`userContentEdits` (`UserContentEditConnection`)	A list of edits to this content. **Arguments for `userContentEdits`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`viewerCanDelete` (`Boolean!`)	Check if the current viewer can delete this object.
`viewerCanMinimize` (`Boolean!`)	Check if the current viewer can minimize this object.
`viewerCanUpdate` (`Boolean!`)	Check if the current viewer can update this object.
`viewerCannotUpdateReasons` (`[CommentCannotUpdateReason!]!`)	Reasons why the current viewer can not update this comment.
`viewerDidAuthor` (`Boolean!`)	Did the viewer author this comment.

GistCommentConnection

The connection type for GistComment.

Fields for `GistCommentConnection`

Name	Description
`edges` (`[GistCommentEdge]`)	A list of edges.
`nodes` (`[GistComment]`)	A list of nodes.

`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

GistCommentEdge

An edge in a connection.

Fields for `GistCommentEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`GistComment`)	The item at the end of the edge.

GistConnection

The connection type for Gist.

Fields for `GistConnection`

Name	Description
`edges` (`[GistEdge]`)	A list of edges.
`nodes` (`[Gist]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

GistEdge

An edge in a connection.

Fields for `GistEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Gist`)	The item at the end of the edge.

GistFile

A file in a gist.

Fields for `GistFile`

Name	Description
`encodedName` (`String`)	The file name encoded to remove characters that are invalid in URL paths.
`encoding` (`String`)	The gist file encoding.
`extension` (`String`)	The file extension from the file name.
`isImage` (`Boolean!`)	Indicates if this file is an image.
`isTruncated` (`Boolean!`)	Whether the file's contents were truncated.
`language` (`Language`)	The programming language this file is written in.
`name` (`String`)	The gist file name.
`size` (`Int`)	The gist file size in bytes.
`text` (`String`)	UTF8 text data or null if the file is binary. **Arguments for `text`** `truncate` (`Int`) Optionally truncate the returned file to this length.

GitActor

Represents an actor in a Git commit (ie. an author or committer).

Fields for `GitActor`

Name	Description
`avatarUrl` (`URI!`)	A URL pointing to the author's public avatar. **Arguments for `avatarUrl`** `size` (`Int`) The size of the resulting square image.
`date` (`GitTimestamp`)	The timestamp of the Git action (authoring or committing).
`email` (`String`)	The email in the Git commit.
`name` (`String`)	The name in the Git commit.
`user` (`User`)	The GitHub user corresponding to the email field. Null if no such user exists.

GitActorConnection

The connection type for GitActor.

Fields for `GitActorConnection`

Name	Description
`edges` (`[GitActorEdge]`)	A list of edges.
`nodes` (`[GitActor]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

GitActorEdge

An edge in a connection.

Fields for `GitActorEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`GitActor`)	The item at the end of the edge.

GitHubMetadata

Represents information about the GitHub instance.

Fields for `GitHubMetadata`

Name	Description
`gitHubServicesSha` (`GitObjectID!`)	Returns a String that's a SHA of `github-services`.
`gitIpAddresses` (`[String!]`)	IP addresses that users connect to for git operations.
`githubEnterpriseImporterIpAddresses` (`[String!]`)	IP addresses that GitHub Enterprise Importer uses for outbound connections.
`hookIpAddresses` (`[String!]`)	IP addresses that service hooks are sent from.
`importerIpAddresses` (`[String!]`)	IP addresses that the importer connects from.
`isPasswordAuthenticationVerifiable` (`Boolean!`)	Whether or not users are verified.
`pagesIpAddresses` (`[String!]`)	IP addresses for GitHub Pages' A records.

GpgSignature

Represents a GPG signature on a Commit or Tag.

`GpgSignature` Implements

- `GitSignature`

Fields for `GpgSignature`

Name	Description
`email` (`String!`)	Email used to sign this object.
`isValid` (`Boolean!`)	True if the signature is valid and verified by GitHub.
`keyId` (`String`)	Hex-encoded ID of the key that signed this object.
`payload` (`String!`)	Payload for GPG signing object. Raw ODB object without the signature header.
`signature` (`String!`)	ASCII-armored signature header from object.
`signer` (`User`)	GitHub user corresponding to the email signing this commit.
`state` (`GitSignatureState!`)	The state of this signature. `VALID` if signature is valid and verified by GitHub, otherwise represents reason why signature is considered invalid.
`wasSignedByGitHub` (`Boolean!`)	True if the signature was made with GitHub's signing key.

HeadRefDeletedEvent

Represents a `head_ref_deleted` event on a given pull request.

`HeadRefDeletedEvent` Implements

- `Node`

Fields for `HeadRefDeletedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`headRef` (`Ref`)	Identifies the Ref associated with the `head_ref_deleted` event.
`headRefName` (`String!`)	Identifies the name of the Ref associated with the `head_ref_deleted` event.
`pullRequest` (`PullRequest!`)	PullRequest referenced by event.

`pullRequest` (`PullRequest!`)	PullRequest referenced by event.

HeadRefForcePushedEvent

Represents a `head_ref_force_pushed` event on a given pull request.

`HeadRefForcePushedEvent` Implements

- `Node`

Fields for `HeadRefForcePushedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`afterCommit` (`Commit`)	Identifies the after commit SHA for the `head_ref_force_pushed` event.
`beforeCommit` (`Commit`)	Identifies the before commit SHA for the `head_ref_force_pushed` event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`pullRequest` (`PullRequest!`)	PullRequest referenced by event.
`ref` (`Ref`)	Identifies the fully qualified ref name for the `head_ref_force_pushed` event.

HeadRefRestoredEvent

Represents a `head_ref_restored` event on a given pull request.

`HeadRefRestoredEvent` Implements

- `Node`

Fields for `HeadRefRestoredEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`pullRequest` (`PullRequest!`)	PullRequest referenced by event.

Hovercard

Detail needed to display a hovercard for a user.

Fields for `Hovercard`

Name	Description
`contexts` (`[HovercardContext!]!`)	Each of the contexts for this hovercard.

IpAllowListEntry

An IP address or range of addresses that is allowed to access an owner's resources.

`IpAllowListEntry` Implements

- `Node`

Fields for `IpAllowListEntry`

Name	Description
`allowListValue` (`String!`)	A single IP address or range of IP addresses in CIDR notation.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`isActive` (`Boolean!`)	Whether the entry is currently active.
`name` (`String`)	The name of the IP allow list entry.
`owner` (`IpAllowListOwner!`)	The owner of the IP allow list entry.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

IpAllowListEntryConnection

The connection type for IpAllowListEntry.

Fields for `IpAllowListEntryConnection`

Name	Description
`edges` (`[IpAllowListEntryEdge]`)	A list of edges.
`nodes` (`[IpAllowListEntry]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

IpAllowListEntryEdge

An edge in a connection.

Fields for `IpAllowListEntryEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`IpAllowListEntry`)	The item at the end of the edge.

Issue

An Issue is a place to discuss ideas, enhancements, tasks, and bugs for a project.

`Issue` Implements

- `Assignable`
- `Closable`
- `Comment`
- `Deletable`
- `Labelable`
- `Lockable`
- `Node`
- `ProjectV2Owner`
- `Reactable`
- `RepositoryNode`
- `Subscribable`
- `SubscribableThread`
- `UniformResourceLocatable`
- `Updatable`
- `UpdatableComment`

Fields for `Issue`

Name	Description
`activeLockReason` (`LockReason`)	Reason that the conversation was locked.
`assignees` (`UserConnection!`)	A list of Users assigned to this object. **Arguments for `assignees`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.

`author` (`Actor`)	The actor who authored the comment.
`authorAssociation` (`CommentAuthorAssociation!`)	Author's association with the subject of the comment.
`body` (`String!`)	Identifies the body of the issue.
`bodyHTML` (`HTML!`)	The body rendered to HTML.
`bodyResourcePath` (`URI!`)	The http path for this issue body.
`bodyText` (`String!`)	Identifies the body of the issue rendered to text.
`bodyUrl` (`URI!`)	The http URL for this issue body.
`closed` (`Boolean!`)	Indicates if the object is closed (definition of closed may depend on type).
`closedAt` (`DateTime`)	Identifies the date and time when the object was closed.
`comments` (`IssueCommentConnection!`)	A list of comments associated with the Issue. **Arguments for `comments`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`IssueCommentOrder`) Ordering options for issue comments returned from the connection.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`createdViaEmail` (`Boolean!`)	Check if this comment was created via an email reply.
`databaseId` (`Int`)	Identifies the primary key from the database.
`editor` (`Actor`)	The actor who edited the comment.
`fullDatabaseId` (`BigInt`)	Identifies the primary key from the database as a BigInt.
`hovercard` (`Hovercard!`)	The hovercard information for this issue.

Arguments for `hovercard`

`includeNotificationContexts` (`Boolean`)
Whether or not to include notification contexts.
The default value is `true`.

`includesCreatedEdit` (`Boolean!`)	Check if this comment was edited and includes an edit with the creation data.
`isPinned` (`Boolean`)	Indicates whether or not this issue is currently pinned to the repository issues list.
`isReadByViewer` (`Boolean`)	Is this issue read by the viewer.
`labels` (`LabelConnection`)	A list of labels associated with the object.

Arguments for `labels`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`LabelOrder`)
Ordering options for labels returned from the connection.

`lastEditedAt` (`DateTime`)	The moment the editor made the last edit.
`linkedBranches` (`LinkedBranchConnection!`)	Branches linked to this issue.

Arguments for `linkedBranches`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`locked` (`Boolean!`)	`true` if the object is locked

`locked` (`Boolean!`)	`true` if the object is locked.
`milestone` (`Milestone`)	Identifies the milestone associated with the issue.
`number` (`Int!`)	Identifies the issue number.
`participants` (`UserConnection!`)	A list of Users that are participating in the Issue conversation. **Arguments for `participants`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`projectCards` (`ProjectCardConnection!`)	List of project cards associated with this issue. **Arguments for `projectCards`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `archivedStates` (`[ProjectCardArchivedState]`) A list of archived states to filter the cards by. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`projectItems` (`ProjectV2ItemConnection!`)	List of project items associated with this issue. **Arguments for `projectItems`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`includeArchived` (`Boolean`)
Include archived items.
The default value is `true` .

`last` (`Int`)
Returns the last *n* elements from the list.

`projectV2` (`ProjectV2`)

Find a project by number.

Arguments for `projectV2`

`number` (`Int!`)
The project number.

`projectsV2` (`ProjectV2Connection!`)

A list of projects under the owner.

Arguments for `projectsV2`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ProjectV2Order`)
How to order the returned projects.

`query` (`String`)
A project to search for under the the owner.

`publishedAt` (`DateTime`)

Identifies when the comment was published at.

`reactionGroups` (`[ReactionGroup!]`)

A list of reactions grouped by content left on the subject.

`reactions` (`ReactionConnection!`)

A list of Reactions left on the Issue.

Arguments for `reactions`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)

Returns the elements in the list that come before the specified cursor.

`content` (`ReactionContent`)
Allows filtering Reactions by emoji.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ReactionOrder`)
Allows specifying the order in which reactions are returned.

`repository` (`Repository!`)	The repository associated with this node.
`resourcePath` (`URI!`)	The HTTP path for this issue.
`state` (`IssueState!`)	Identifies the state of the issue.
`stateReason` (`IssueStateReason`)	Identifies the reason for the issue state.
`timeline` (`IssueTimelineConnection!`)	A list of events, comments, commits, etc. associated with the issue.

Deprecation notice
`timeline` is deprecated.
`timeline` will be removed Use Issue.timelineItems instead. Removal on 2020-10-01 UTC.

Arguments for `timeline`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`since` (`DateTime`)
Allows filtering timeline events by a `since` timestamp.

`timelineItems` (`IssueTimelineItemsConnection!`)	A list of events, comments, commits, etc. associated with the issue.

Arguments for `timelineItems`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`itemTypes` (`[IssueTimelineItemsItemType!]`)
Filter timeline items by type.

`last` (`Int`)
Returns the last *n* elements from the list.

`since` (`DateTime`)
Filter timeline items by a `since` timestamp.

`skip` (`Int`)
Skips the first *n* elements in the list.

`title` (`String!`)	Identifies the issue title.
`titleHTML` (`String!`)	Identifies the issue title rendered to HTML.
`trackedInIssues` (`IssueConnection!`)	A list of issues that track this issue.

Arguments for `trackedInIssues`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`trackedIssues` (`IssueConnection!`)	A list of issues tracked inside the current issue.

Arguments for `trackedIssues`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

	`last` (`Int`) Returns the last *n* elements from the list.
`trackedIssuesCount` (`Int!`)	The number of tracked issues for this issue. **Arguments for `trackedIssuesCount`** `states` (`[TrackedIssueStates]`) Limit the count to tracked issues with the specified states.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this issue.
`userContentEdits` (`UserContentEditConnection`)	A list of edits to this content. **Arguments for `userContentEdits`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`viewerCanClose` (`Boolean!`)	Indicates if the object can be closed by the viewer.
`viewerCanDelete` (`Boolean!`)	Check if the current viewer can delete this object.
`viewerCanReact` (`Boolean!`)	Can user react to this subject.
`viewerCanReopen` (`Boolean!`)	Indicates if the object can be reopened by the viewer.
`viewerCanSubscribe` (`Boolean!`)	Check if the viewer is able to change their subscription status for the repository.
`viewerCanUpdate` (`Boolean!`)	Check if the current viewer can update this object.
`viewerCannotUpdateReasons` (`[CommentCannotUpdateReason!]!`)	Reasons why the current viewer can not update this comment.
`viewerDidAuthor` (`Boolean!`)	Did the viewer author this comment.

`viewerSubscription` (`SubscriptionState`)	Identifies if the viewer is watching, not watching, or ignoring the subscribable entity.
`viewerThreadSubscriptionFormAction` (`ThreadSubscriptionFormAction`)	Identifies the viewer's thread subscription form action.
`viewerThreadSubscriptionStatus` (`ThreadSubscriptionState`)	Identifies the viewer's thread subscription status.

IssueComment

Represents a comment on an Issue.

`IssueComment` Implements

- `Comment`
- `Deletable`
- `Minimizable`
- `Node`
- `Reactable`
- `RepositoryNode`
- `Updatable`
- `UpdatableComment`

Fields for `IssueComment`

Name	Description
`author` (`Actor`)	The actor who authored the comment.
`authorAssociation` (`CommentAuthorAssociation!`)	Author's association with the subject of the comment.
`body` (`String!`)	The body as Markdown.
`bodyHTML` (`HTML!`)	The body rendered to HTML.
`bodyText` (`String!`)	The body rendered to text.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`createdViaEmail` (`Boolean!`)	Check if this comment was created via an email reply.
`databaseId` (`Int`)	Identifies the primary key from the database.
`editor` (`Actor`)	The actor who edited the comment.
`fullDatabaseId` (`BigInt`)	Identifies the primary key from the database as a BigInt.
`includesCreatedEdit` (`Boolean!`)	Check if this comment was edited and includes an edit with the creation data.
`isMinimized` (`Boolean!`)	Returns whether or not a comment has been minimized.

`issue` (`Issue!`)	Identifies the issue associated with the comment.
`lastEditedAt` (`DateTime`)	The moment the editor made the last edit.
`minimizedReason` (`String`)	Returns why the comment was minimized. One of `abuse` , `off-topic` , `outdated` , `resolved` , `duplicate` and `spam` . Note that the case and formatting of these values differs from the inputs to the `MinimizeComment` mutation.
`publishedAt` (`DateTime`)	Identifies when the comment was published at.
`pullRequest` (`PullRequest`)	Returns the pull request associated with the comment, if this comment was made on a pull request.
`reactionGroups` (`[ReactionGroup!]`)	A list of reactions grouped by content left on the subject.
`reactions` (`ReactionConnection!`)	A list of Reactions left on the Issue. **Arguments for `reactions`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `content` (`ReactionContent`) Allows filtering Reactions by emoji. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`ReactionOrder`) Allows specifying the order in which reactions are returned.
`repository` (`Repository!`)	The repository associated with this node.
`resourcePath` (`URI!`)	The HTTP path for this issue comment.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this issue comment.
`userContentEdits` (`UserContentEditConnection`)	A list of edits to this content. **Arguments for `userContentEdits`** `after` (`String`) Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`viewerCanDelete` (`Boolean!`)	Check if the current viewer can delete this object.
`viewerCanMinimize` (`Boolean!`)	Check if the current viewer can minimize this object.
`viewerCanReact` (`Boolean!`)	Can user react to this subject.
`viewerCanUpdate` (`Boolean!`)	Check if the current viewer can update this object.
`viewerCannotUpdateReasons` (`[CommentCannotUpdateReason!]!`)	Reasons why the current viewer can not update this comment.
`viewerDidAuthor` (`Boolean!`)	Did the viewer author this comment.

IssueCommentConnection

The connection type for IssueComment.

Fields for `IssueCommentConnection`

Name	Description
`edges` (`[IssueCommentEdge]`)	A list of edges.
`nodes` (`[IssueComment]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

IssueCommentEdge

An edge in a connection.

Fields for `IssueCommentEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`IssueComment`)	The item at the end of the edge.

IssueConnection

The connection type for Issue.

Fields for `IssueConnection`

Name	Description
`edges` (`[IssueEdge]`)	A list of edges.
`nodes` (`[Issue]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

IssueContributionsByRepository

This aggregates issues opened by a user within one repository.

Fields for `IssueContributionsByRepository`

Name	Description
`contributions` (`CreatedIssueContributionConnection!`)	The issue contributions. **Arguments for `contributions`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`ContributionOrder`) Ordering options for contributions returned from the connection.
`repository` (`Repository!`)	The repository in which the issues were opened.

IssueEdge

An edge in a connection.

Fields for `IssueEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Issue`)	The item at the end of the edge.

IssueTemplate

A repository issue template.

Fields for `IssueTemplate`

Name	Description
`about` (`String`)	The template purpose.
`assignees` (`UserConnection!`)	The suggested assignees. **Arguments for `assignees`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`body` (`String`)	The suggested issue body.
`filename` (`String!`)	The template filename.
`labels` (`LabelConnection`)	The suggested issue labels. **Arguments for `labels`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.

	`orderBy` (`LabelOrder`) Ordering options for labels returned from the connection.
`name` (`String!`)	The template name.
`title` (`String`)	The suggested issue title.

IssueTimelineConnection

The connection type for IssueTimelineItem.

Fields for `IssueTimelineConnection`

Name	Description
`edges` (`[IssueTimelineItemEdge]`)	A list of edges.
`nodes` (`[IssueTimelineItem]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

IssueTimelineItemEdge

An edge in a connection.

Fields for `IssueTimelineItemEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`IssueTimelineItem`)	The item at the end of the edge.

IssueTimelineItemsConnection

The connection type for IssueTimelineItems.

Fields for `IssueTimelineItemsConnection`

Name	Description
`edges` (`[IssueTimelineItemsEdge]`)	A list of edges.
`filteredCount` (`Int!`)	Identifies the count of items after applying `before` and `after` filters.
`nodes` (`[IssueTimelineItems]`)	A list of nodes.

`pageCount` (`Int!`)	Identifies the count of items after applying `before` / `after` filters and `first` / `last` / `skip` slicing.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.
`updatedAt` (`DateTime!`)	Identifies the date and time when the timeline was last updated.

IssueTimelineItemsEdge

An edge in a connection.

Fields for `IssueTimelineItemsEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`IssueTimelineItems`)	The item at the end of the edge.

JoinedGitHubContribution

Represents a user signing up for a GitHub account.

`JoinedGitHubContribution` Implements

- `Contribution`

Fields for `JoinedGitHubContribution`

Name	Description
`isRestricted` (`Boolean!`)	Whether this contribution is associated with a record you do not have access to. For example, your own 'first issue' contribution may have been made on a repository you can no longer access.
`occurredAt` (`DateTime!`)	When this contribution was made.
`resourcePath` (`URI!`)	The HTTP path for this contribution.
`url` (`URI!`)	The HTTP URL for this contribution.
`user` (`User!`)	The user who made this contribution.

Label

A label for categorizing Issues, Pull Requests, Milestones, or Discussions with a given

Repository.

`Label` Implements

- `Node`

Fields for `Label`

Name	Description
`color` (`String!`)	Identifies the label color.
`createdAt` (`DateTime`)	Identifies the date and time when the label was created.
`description` (`String`)	A brief description of this label.
`isDefault` (`Boolean!`)	Indicates whether or not this is a default label.
`issues` (`IssueConnection!`)	A list of issues associated with this label. **Arguments for `issues`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `filterBy` (`IssueFilters`) Filtering options for issues returned from the connection. `first` (`Int`) Returns the first *n* elements from the list. `labels` (`[String!]`) A list of label names to filter the pull requests by. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`IssueOrder`) Ordering options for issues returned from the connection. `states` (`[IssueState!]`) A list of states to filter the issues by.
`name` (`String!`)	Identifies the label name.
`pullRequests` (`PullRequestConnection!`)	A list of pull requests associated with this label. **Arguments for `pullRequests`** `after` (`String`) Returns the elements in the list that come

Returns the elements in the list that come after the specified cursor.

`baseRefName` (`String`)
The base ref name to filter the pull requests by.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`headRefName` (`String`)
The head ref name to filter the pull requests by.

`labels` (`[String!]`)
A list of label names to filter the pull requests by.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`IssueOrder`)
Ordering options for pull requests returned from the connection.

`states` (`[PullRequestState!]`)
A list of states to filter the pull requests by.

`repository` (`Repository!`)	The repository associated with this label.
`resourcePath` (`URI!`)	The HTTP path for this label.
`updatedAt` (`DateTime`)	Identifies the date and time when the label was last updated.
`url` (`URI!`)	The HTTP URL for this label.

LabelConnection

The connection type for Label.

Fields for `LabelConnection`

Name	Description
`edges` (`[LabelEdge]`)	A list of edges.
`nodes` (`[Label]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

LabelEdge

An edge in a connection.

Fields for `LabelEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Label`)	The item at the end of the edge.

LabeledEvent

Represents a `labeled` event on a given issue or pull request.

`LabeledEvent` Implements

- `Node`

Fields for `LabeledEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`label` (`Label!`)	Identifies the label associated with the `labeled` event.
`labelable` (`Labelable!`)	Identifies the `Labelable` associated with the event.

Language

Represents a given language found in repositories.

`Language` Implements

- `Node`

Fields for `Language`

Name	Description
`color` (`String`)	The color defined for the current language.
`name` (`String!`)	The name of the current language.

LanguageConnection

A list of languages associated with the parent.

Fields for `LanguageConnection`

Name	Description
`edges` (`[LanguageEdge]`)	A list of edges.
`nodes` (`[Language]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.
`totalSize` (`Int!`)	The total size in bytes of files written in that language.

LanguageEdge

Represents the language of a repository.

Fields for `LanguageEdge`

Name	Description
`size` (`Int!`)	The number of bytes of code written in the language.

License

A repository's open source license.

`License` Implements

- `Node`

Fields for `License`

Name	Description
`body` (`String!`)	The full text of the license.
`conditions` (`[LicenseRule]!`)	The conditions set by the license.
`description` (`String`)	A human-readable description of the license.
`featured` (`Boolean!`)	Whether the license should be featured.
`hidden` (`Boolean!`)	Whether the license should be displayed in license pickers.
`implementation` (`String`)	Instructions on how to implement the license.
`key` (`String!`)	The lowercased SPDX ID of the license.

`limitations` (`[LicenseRule]!`)	The limitations set by the license.
`name` (`String!`)	The license full name specified by https://spdx.org/licenses.
`nickname` (`String`)	Customary short name if applicable (e.g, GPLv3).
`permissions` (`[LicenseRule]!`)	The permissions set by the license.
`pseudoLicense` (`Boolean!`)	Whether the license is a pseudo-license placeholder (e.g., other, no-license).
`spdxId` (`String`)	Short identifier specified by https://spdx.org/licenses.
`url` (`URI`)	URL to the license on https://choosealicense.com.

LicenseRule

Describes a License's conditions, permissions, and limitations.

Fields for `LicenseRule`

Name	Description
`description` (`String!`)	A description of the rule.
`key` (`String!`)	The machine-readable rule key.
`label` (`String!`)	The human-readable rule label.

LinkedBranch

A branch linked to an issue.

`LinkedBranch` Implements

- `Node`

Fields for `LinkedBranch`

Name	Description
`ref` (`Ref`)	The branch's ref.

LinkedBranchConnection

The connection type for LinkedBranch.

Fields for `LinkedBranchConnection`

Name	Description
`edges` (`[LinkedBranchEdge]`)	A list of edges.
`nodes` (`[LinkedBranch]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

LinkedBranchEdge

An edge in a connection.

Fields for `LinkedBranchEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`LinkedBranch`)	The item at the end of the edge.

LockedEvent

Represents a `locked` event on a given issue or pull request.

`LockedEvent` Implements

- `Node`

Fields for `LockedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`lockReason` (`LockReason`)	Reason that the conversation was locked (optional).
`lockable` (`Lockable!`)	Object that was locked.

Mannequin

A placeholder user for attribution of imported data on GitHub.

`Mannequin` Implements

- `Actor`
- `Node`

- `UniformResourceLocatable`

Fields for `Mannequin`

Name	Description
`avatarUrl` (`URI!`)	A URL pointing to the GitHub App's public avatar. **Arguments for `avatarUrl`** `size` (`Int`) The size of the resulting square image.
`claimant` (`User`)	The user that has claimed the data attributed to this mannequin.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`email` (`String`)	The mannequin's email on the source instance.
`login` (`String!`)	The username of the actor.
`resourcePath` (`URI!`)	The HTML path to this resource.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The URL to this resource.

MannequinConnection

The connection type for Mannequin.

Fields for `MannequinConnection`

Name	Description
`edges` (`[MannequinEdge]`)	A list of edges.
`nodes` (`[Mannequin]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

MannequinEdge

Represents a mannequin.

Fields for `MannequinEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Mannequin`)	The item at the end of the edge.

MarkedAsDuplicateEvent

Represents a `marked_as_duplicate` event on a given issue or pull request.

`MarkedAsDuplicateEvent` Implements

- `Node`

Fields for `MarkedAsDuplicateEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`canonical` (`IssueOrPullRequest`)	The authoritative issue or pull request which has been duplicated by another.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`duplicate` (`IssueOrPullRequest`)	The issue or pull request which has been marked as a duplicate of another.
`isCrossRepository` (`Boolean!`)	Canonical and duplicate belong to different repositories.

MarketplaceCategory

A public description of a Marketplace category.

`MarketplaceCategory` Implements

- `Node`

Fields for `MarketplaceCategory`

Name	Description
`description` (`String`)	The category's description.
`howItWorks` (`String`)	The technical description of how apps listed in this category work with GitHub.
`name` (`String!`)	The category's name.
`primaryListingCount` (`Int!`)	How many Marketplace listings have this as their primary category.

`resourcePath` (`URI!`)	The HTTP path for this Marketplace category.
`secondaryListingCount` (`Int!`)	How many Marketplace listings have this as their secondary category.
`slug` (`String!`)	The short name of the category used in its URL.
`url` (`URI!`)	The HTTP URL for this Marketplace category.

MarketplaceListing

A listing in the GitHub integration marketplace.

`MarketplaceListing` Implements

- `Node`

Fields for `MarketplaceListing`

Name	Description
`app` (`App`)	The GitHub App this listing represents.
`companyUrl` (`URI`)	URL to the listing owner's company site.
`configurationResourcePath` (`URI!`)	The HTTP path for configuring access to the listing's integration or OAuth app.
`configurationUrl` (`URI!`)	The HTTP URL for configuring access to the listing's integration or OAuth app.
`documentationUrl` (`URI`)	URL to the listing's documentation.
`extendedDescription` (`String`)	The listing's detailed description.
`extendedDescriptionHTML` (`HTML!`)	The listing's detailed description rendered to HTML.
`fullDescription` (`String!`)	The listing's introductory description.
`fullDescriptionHTML` (`HTML!`)	The listing's introductory description rendered to HTML.
`hasPublishedFreeTrialPlans` (`Boolean!`)	Does this listing have any plans with a free trial?.
`hasTermsOfService` (`Boolean!`)	Does this listing have a terms of service link?.
`hasVerifiedOwner` (`Boolean!`)	Whether the creator of the app is a verified org.
`howItWorks` (`String`)	A technical description of how this app works with GitHub.
`howItWorksHTML` (`HTML!`)	The listing's technical description rendered to HTML.
`installationUrl` (`URI`)	URL to install the product to the viewer's account or organization.

account or organization.

`installedForViewer` (`Boolean!`)	Whether this listing's app has been installed for the current viewer.
`isArchived` (`Boolean!`)	Whether this listing has been removed from the Marketplace.
`isDraft` (`Boolean!`)	Whether this listing is still an editable draft that has not been submitted for review and is not publicly visible in the Marketplace.
`isPaid` (`Boolean!`)	Whether the product this listing represents is available as part of a paid plan.
`isPublic` (`Boolean!`)	Whether this listing has been approved for display in the Marketplace.
`isRejected` (`Boolean!`)	Whether this listing has been rejected by GitHub for display in the Marketplace.
`isUnverified` (`Boolean!`)	Whether this listing has been approved for unverified display in the Marketplace.
`isUnverifiedPending` (`Boolean!`)	Whether this draft listing has been submitted for review for approval to be unverified in the Marketplace.
`isVerificationPendingFromDraft` (`Boolean!`)	Whether this draft listing has been submitted for review from GitHub for approval to be verified in the Marketplace.
`isVerificationPendingFromUnverified` (`Boolean!`)	Whether this unverified listing has been submitted for review from GitHub for approval to be verified in the Marketplace.
`isVerified` (`Boolean!`)	Whether this listing has been approved for verified display in the Marketplace.
`logoBackgroundColor` (`String!`)	The hex color code, without the leading '#', for the logo background.
`logoUrl` (`URI`)	URL for the listing's logo image. **Arguments for `logoUrl`** `size` (`Int`) The size in pixels of the resulting square image. The default value is `400`.
`name` (`String!`)	The listing's full name.
`normalizedShortDescription` (`String!`)	The listing's very short description without a trailing period or ampersands.
`pricingUrl` (`URI`)	URL to the listing's detailed pricing.
`primaryCategory` (`MarketplaceCategory!`)	The category that best describes the listing.
`privacyPolicyUrl` (`URI!`)	URL to the listing's privacy policy, may return an empty string for listings that do not require a

	privacy policy URL.
`resourcePath` (`URI!`)	The HTTP path for the Marketplace listing.
`screenshotUrls` (`[String]!`)	The URLs for the listing's screenshots.
`secondaryCategory` (`MarketplaceCategory`)	An alternate category that describes the listing.
`shortDescription` (`String!`)	The listing's very short description.
`slug` (`String!`)	The short name of the listing used in its URL.
`statusUrl` (`URI`)	URL to the listing's status page.
`supportEmail` (`String`)	An email address for support for this listing's app.
`supportUrl` (`URI!`)	Either a URL or an email address for support for this listing's app, may return an empty string for listings that do not require a support URL.
`termsOfServiceUrl` (`URI`)	URL to the listing's terms of service.
`url` (`URI!`)	The HTTP URL for the Marketplace listing.
`viewerCanAddPlans` (`Boolean!`)	Can the current viewer add plans for this Marketplace listing.
`viewerCanApprove` (`Boolean!`)	Can the current viewer approve this Marketplace listing.
`viewerCanDelist` (`Boolean!`)	Can the current viewer delist this Marketplace listing.
`viewerCanEdit` (`Boolean!`)	Can the current viewer edit this Marketplace listing.
`viewerCanEditCategories` (`Boolean!`)	Can the current viewer edit the primary and secondary category of this Marketplace listing.
`viewerCanEditPlans` (`Boolean!`)	Can the current viewer edit the plans for this Marketplace listing.
`viewerCanRedraft` (`Boolean!`)	Can the current viewer return this Marketplace listing to draft state so it becomes editable again.
`viewerCanReject` (`Boolean!`)	Can the current viewer reject this Marketplace listing by returning it to an editable draft state or rejecting it entirely.
`viewerCanRequestApproval` (`Boolean!`)	Can the current viewer request this listing be reviewed for display in the Marketplace as verified.
`viewerHasPurchased` (`Boolean!`)	Indicates whether the current user has an active subscription to this Marketplace listing.
`viewerHasPurchasedForAllOrganizations` (`Boolean!`)	Indicates if the current user has purchased a subscription to this Marketplace listing for all of the organizations the user owns.
`viewerIsListingAdmin` (`Boolean!`)	Does the current viewer role allow them to administer this Marketplace listing.

MarketplaceListingConnection

Look up Marketplace Listings.

Fields for `MarketplaceListingConnection`

Name	Description
`edges` (`[MarketplaceListingEdge]`)	A list of edges.
`nodes` (`[MarketplaceListing]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

MarketplaceListingEdge

An edge in a connection.

Fields for `MarketplaceListingEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`MarketplaceListing`)	The item at the end of the edge.

MembersCanDeleteReposClearAuditEntry

Audit log entry for a members_can_delete_repos.clear event.

`MembersCanDeleteReposClearAuditEntry` Implements

- `AuditEntry`
- `EnterpriseAuditEntryData`
- `Node`
- `OrganizationAuditEntryData`

Fields for `MembersCanDeleteReposClearAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.

`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`enterpriseResourcePath` (`URI`)	The HTTP path for this enterprise.
`enterpriseSlug` (`String`)	The slug of the enterprise.
`enterpriseUrl` (`URI`)	The HTTP URL for this enterprise.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

MembersCanDeleteReposDisableAuditEntry

Audit log entry for a members_can_delete_repos.disable event.

`MembersCanDeleteReposDisableAuditEntry` Implements

- `AuditEntry`
- `EnterpriseAuditEntryData`
- `Node`
- `OrganizationAuditEntryData`

Fields for `MembersCanDeleteReposDisableAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.

	location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`enterpriseResourcePath` (`URI`)	The HTTP path for this enterprise.
`enterpriseSlug` (`String`)	The slug of the enterprise.
`enterpriseUrl` (`URI`)	The HTTP URL for this enterprise.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

MembersCanDeleteReposEnableAuditEntry

Audit log entry for a members_can_delete_repos.enable event.

`MembersCanDeleteReposEnableAuditEntry` Implements

- `AuditEntry`
- `EnterpriseAuditEntryData`
- `Node`
- `OrganizationAuditEntryData`

Fields for `MembersCanDeleteReposEnableAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's

	location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`enterpriseResourcePath` (`URI`)	The HTTP path for this enterprise.
`enterpriseSlug` (`String`)	The slug of the enterprise.
`enterpriseUrl` (`URI`)	The HTTP URL for this enterprise.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

MentionedEvent

Represents a `mentioned` event on a given issue or pull request.

`MentionedEvent` Implements

- `Node`

Fields for `MentionedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.

MergeQueue

The queue of pull request entries to be merged into a protected branch in a repository.

`MergeQueue` Implements

- `Node`

Fields for `MergeQueue`

Name	Description
`configuration` (`MergeQueueConfiguration`)	The configuration for this merge queue.
`entries` (`MergeQueueEntryConnection`)	The entries in the queue. **Arguments for `entries`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`nextEntryEstimatedTimeToMerge` (`Int`)	The estimated time in seconds until a newly added entry would be merged.
`repository` (`Repository`)	The repository this merge queue belongs to.
`resourcePath` (`URI!`)	The HTTP path for this merge queue.
`url` (`URI!`)	The HTTP URL for this merge queue.

MergeQueueConfiguration

Configuration for a MergeQueue.

Fields for `MergeQueueConfiguration`

Name	Description
`checkResponseTimeout` (`Int`)	The amount of time in minutes to wait for a check response before considering it a failure.
`maximumEntriesToBuild` (`Int`)	The maximum number of entries to build at once.
`maximumEntriesToMerge` (`Int`)	The maximum number of entries to merge at

`maximumEntriesToMerge` (`Int`)	The maximum number of entries to merge at once.
`mergeMethod` (`PullRequestMergeMethod`)	The merge method to use for this queue.
`mergingStrategy` (`MergeQueueMergingStrategy`)	The strategy to use when merging entries.
`minimumEntriesToMerge` (`Int`)	The minimum number of entries required to merge at once.
`minimumEntriesToMergeWaitTime` (`Int`)	The amount of time in minutes to wait before ignoring the minumum number of entries in the queue requirement and merging a collection of entries.

MergeQueueEntry

Entries in a MergeQueue.

`MergeQueueEntry` Implements

- `Node`

Fields for `MergeQueueEntry`

Name	Description
`baseCommit` (`Commit`)	The base commit for this entry.
`enqueuedAt` (`DateTime!`)	The date and time this entry was added to the merge queue.
`enqueuer` (`Actor!`)	The actor that enqueued this entry.
`estimatedTimeToMerge` (`Int`)	The estimated time in seconds until this entry will be merged.
`headCommit` (`Commit`)	The head commit for this entry.
`jump` (`Boolean!`)	Whether this pull request should jump the queue.
`mergeQueue` (`MergeQueue`)	The merge queue that this entry belongs to.
`position` (`Int!`)	The position of this entry in the queue.
`pullRequest` (`PullRequest`)	The pull request that will be added to a merge group.
`solo` (`Boolean!`)	Does this pull request need to be deployed on its own.
`state` (`MergeQueueEntryState!`)	The state of this entry in the queue.

MergeQueueEntryConnection

The connection type for MergeQueueEntry.

Fields for `MergeQueueEntryConnection`

Name	Description
`edges` (`[MergeQueueEntryEdge]`)	A list of edges.
`nodes` (`[MergeQueueEntry]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

MergeQueueEntryEdge

An edge in a connection.

Fields for `MergeQueueEntryEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`MergeQueueEntry`)	The item at the end of the edge.

MergedEvent

Represents a `merged` event on a given pull request.

`MergedEvent` Implements

- `Node`
- `UniformResourceLocatable`

Fields for `MergedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`commit` (`Commit`)	Identifies the commit associated with the `merge` event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`mergeRef` (`Ref`)	Identifies the Ref associated with the `merge` event.
`mergeRefName` (`String!`)	Identifies the name of the Ref associated with the `merge` event.
`pullRequest` (`PullRequest!`)	PullRequest referenced by event.
`resourcePath` (`URI!`)	The HTTP path for this merged event.

`url` (`URI!`)	The HTTP URL for this merged event.

MigrationSource

A GitHub Enterprise Importer (GEI) migration source.

`MigrationSource` Implements

- `Node`

Fields for `MigrationSource`

Name	Description
`name` (`String!`)	The migration source name.
`type` (`MigrationSourceType!`)	The migration source type.
`url` (`URI!`)	The migration source URL, for example `https://github.com` or `https://monalisa.ghe.com`.

Milestone

Represents a Milestone object on a given repository.

`Milestone` Implements

- `Closable`
- `Node`
- `UniformResourceLocatable`

Fields for `Milestone`

Name	Description
`closed` (`Boolean!`)	Indicates if the object is closed (definition of closed may depend on type).
`closedAt` (`DateTime`)	Identifies the date and time when the object was closed.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`creator` (`Actor`)	Identifies the actor who created the milestone.
`description` (`String`)	Identifies the description of the milestone.
`dueOn` (`DateTime`)	Identifies the due date of the milestone.
`issues` (`IssueConnection!`)	A list of issues associated with the milestone.

Arguments for `issues`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`filterBy` (`IssueFilters`)
Filtering options for issues returned from the connection.

`first` (`Int`)
Returns the first *n* elements from the list.

`labels` (`[String!]`)
A list of label names to filter the pull requests by.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`IssueOrder`)
Ordering options for issues returned from the connection.

`states` (`[IssueState!]`)
A list of states to filter the issues by.

`number` (`Int!`)	Identifies the number of the milestone.
`progressPercentage` (`Float!`)	Identifies the percentage complete for the milestone.
`pullRequests` (`PullRequestConnection!`)	A list of pull requests associated with the milestone.

Arguments for `pullRequests`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`baseRefName` (`String`)
The base ref name to filter the pull requests by.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`headRefName` (`String`)
The head ref name to filter the pull requests by.

`labels` (`[String!]`)
A list of label names to filter the pull requests by.

requests by.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`IssueOrder`)
Ordering options for pull requests returned from the connection.

`states` (`[PullRequestState!]`)
A list of states to filter the pull requests by.

`repository` (`Repository!`)	The repository associated with this milestone.
`resourcePath` (`URI!`)	The HTTP path for this milestone.
`state` (`MilestoneState!`)	Identifies the state of the milestone.
`title` (`String!`)	Identifies the title of the milestone.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this milestone.
`viewerCanClose` (`Boolean!`)	Indicates if the object can be closed by the viewer.
`viewerCanReopen` (`Boolean!`)	Indicates if the object can be reopened by the viewer.

MilestoneConnection

The connection type for Milestone.

Fields for `MilestoneConnection`

Name	Description
`edges` (`[MilestoneEdge]`)	A list of edges.
`nodes` (`[Milestone]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

MilestoneEdge

An edge in a connection.

Fields for `MilestoneEdge`

Name	Description

`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Milestone`)	The item at the end of the edge.

MilestonedEvent

Represents a `milestoned` event on a given issue or pull request.

`MilestonedEvent` Implements

- `Node`

Fields for `MilestonedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`milestoneTitle` (`String!`)	Identifies the milestone title associated with the `milestoned` event.
`subject` (`MilestoneItem!`)	Object referenced by event.

MovedColumnsInProjectEvent

Represents a `moved_columns_in_project` event on a given issue or pull request.

`MovedColumnsInProjectEvent` Implements

- `Node`

Fields for `MovedColumnsInProjectEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`previousProjectColumnName` (`String!`)	Column name the issue or pull request was moved from. **Preview notice** `previousProjectColumnName` is available under the Project event details preview. During the preview period, the API may change without notice.

`project` (`Project`)	Project referenced by event. **Preview notice** `project` is available under the Project event details preview. During the preview period, the API may change without notice.
`projectCard` (`ProjectCard`)	Project card referenced by this project event. **Preview notice** `projectCard` is available under the Project event details preview. During the preview period, the API may change without notice.
`projectColumnName` (`String!`)	Column name the issue or pull request was moved to. **Preview notice** `projectColumnName` is available under the Project event details preview. During the preview period, the API may change without notice.

OIDCProvider

An OIDC identity provider configured to provision identities for an enterprise. Visible to enterprise owners or enterprise owners' personal access tokens (classic) with read:enterprise or admin:enterprise scope.

`OIDCProvider` Implements

- `Node`

Fields for `OIDCProvider`

Name	Description
`enterprise` (`Enterprise`)	The enterprise this identity provider belongs to.
`externalIdentities` (`ExternalIdentityConnection!`)	ExternalIdentities provisioned by this identity provider. **Arguments for `externalIdentities`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`login` (`String`)
Filter to external identities with the users login.

`membersOnly` (`Boolean`)
Filter to external identities with valid org membership only.

`userName` (`String`)
Filter to external identities with the users userName/NameID attribute.

`providerType` (`OIDCProviderType!`)	The OIDC identity provider type.
`tenantId` (`String!`)	The id of the tenant this provider is attached to.

OauthApplicationCreateAuditEntry

Audit log entry for a oauth_application.create event.

`OauthApplicationCreateAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OauthApplicationAuditEntryData`
- `OrganizationAuditEntryData`

Fields for `OauthApplicationCreateAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`applicationUrl` (`URI`)	The application URL of the OAuth application.
`callbackUrl` (`URI`)	The callback URL of the OAuth application.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`oauthApplicationName` (`String`)	The name of the OAuth application.

`oauthApplicationName` (`String`)	The name of the OAuth application.
`oauthApplicationResourcePath` (`URI`)	The HTTP path for the OAuth application.
`oauthApplicationUrl` (`URI`)	The HTTP URL for the OAuth application.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`rateLimit` (`Int`)	The rate limit of the OAuth application.
`state` (`OauthApplicationCreateAuditEntryState`)	The state of the OAuth application.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgAddBillingManagerAuditEntry

Audit log entry for a org.add_billing_manager.

`OrgAddBillingManagerAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgAddBillingManagerAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.

`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`invitationEmail` (`String`)	The email address used to invite a billing manager for the organization.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgAddMemberAuditEntry

Audit log entry for a org.add_member.

`OrgAddMemberAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgAddMemberAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.

`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`permission` (`OrgAddMemberAuditEntryPermission`)	The permission level of the member added to the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgBlockUserAuditEntry

Audit log entry for a org.block_user.

`OrgBlockUserAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgBlockUserAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`blockedUser` (`User`)	The blocked user.
`blockedUserName` (`String`)	The username of the blocked user.
`blockedUserResourcePath` (`URI`)	The HTTP path for the blocked user.

`blockedUserUrl` (`URI`)	The HTTP URL for the blocked user.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgConfigDisableCollaboratorsOnlyAuditEntry

Audit log entry for a org.config.disable_collaborators_only event.

`OrgConfigDisableCollaboratorsOnlyAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgConfigDisableCollaboratorsOnlyAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.

`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgConfigEnableCollaboratorsOnlyAuditEntry

Audit log entry for a org.config.enable_collaborators_only event.

`OrgConfigEnableCollaboratorsOnlyAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgConfigEnableCollaboratorsOnlyAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.

`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgCreateAuditEntry

Audit log entry for a org.create event.

`OrgCreateAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgCreateAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`billingPlan` (`OrgCreateAuditEntryBillingPlan`)	The billing plan for the Organization.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.

`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgDisableOauthAppRestrictionsAuditEntry

Audit log entry for a org.disable_oauth_app_restrictions event.

`OrgDisableOauthAppRestrictionsAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgDisableOauthAppRestrictionsAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.

`userUrl` (`URI`)	The HTTP URL for the user.

OrgDisableSamlAuditEntry

Audit log entry for a org.disable_saml event.

`OrgDisableSamlAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgDisableSamlAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`digestMethodUrl` (`URI`)	The SAML provider's digest algorithm URL.
`issuerUrl` (`URI`)	The SAML provider's issuer URL.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`signatureMethodUrl` (`URI`)	The SAML provider's signature algorithm URL.
`singleSignOnUrl` (`URI`)	The SAML provider's single sign-on URL.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.

`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgDisableTwoFactorRequirementAuditEntry

Audit log entry for a org.disable_two_factor_requirement event.

`OrgDisableTwoFactorRequirementAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgDisableTwoFactorRequirementAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgEnableOauthAppRestrictionsAuditEntry

Audit log entry for a org.enable_oauth_app_restrictions event.

`OrgEnableOauthAppRestrictionsAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgEnableOauthAppRestrictionsAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgEnableSamlAuditEntry

Audit log entry for a org.enable_saml event.

`OrgEnableSamlAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgEnableSamlAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`digestMethodUrl` (`URI`)	The SAML provider's digest algorithm URL.
`issuerUrl` (`URI`)	The SAML provider's issuer URL.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`signatureMethodUrl` (`URI`)	The SAML provider's signature algorithm URL.
`singleSignOnUrl` (`URI`)	The SAML provider's single sign-on URL.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgEnableTwoFactorRequirementAuditEntry

Audit log entry for a org.enable_two_factor_requirement event.

`OrgEnableTwoFactorRequirementAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgEnableTwoFactorRequirementAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgInviteMemberAuditEntry

Audit log entry for a org.invite_member event.

`OrgInviteMemberAuditEntry` Implements

- `AuditEntry`

- Node
- OrganizationAuditEntryData

Fields for `OrgInviteMemberAuditEntry`

Name	Description
`action` (String!)	The action name.
`actor` (AuditEntryActor)	The user who initiated the action.
`actorIp` (String)	The IP address of the actor.
`actorLocation` (ActorLocation)	A readable representation of the actor's location.
`actorLogin` (String)	The username of the user who initiated the action.
`actorResourcePath` (URI)	The HTTP path for the actor.
`actorUrl` (URI)	The HTTP URL for the actor.
`createdAt` (PreciseDateTime!)	The time the action was initiated.
`email` (String)	The email address of the organization invitation.
`operationType` (OperationType)	The corresponding operation type for the action.
`organization` (Organization)	The Organization associated with the Audit Entry.
`organizationInvitation` (OrganizationInvitation)	The organization invitation.
`organizationName` (String)	The name of the Organization.
`organizationResourcePath` (URI)	The HTTP path for the organization.
`organizationUrl` (URI)	The HTTP URL for the organization.
`user` (User)	The user affected by the action.
`userLogin` (String)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (URI)	The HTTP path for the user.
`userUrl` (URI)	The HTTP URL for the user.

OrgInviteToBusinessAuditEntry

Audit log entry for a org.invite_to_business event.

`OrgInviteToBusinessAuditEntry` Implements

- AuditEntry

- EnterpriseAuditEntryData
- Node
- OrganizationAuditEntryData

Fields for `OrgInviteToBusinessAuditEntry`

Name	Description
`action` (String!)	The action name.
`actor` (AuditEntryActor)	The user who initiated the action.
`actorIp` (String)	The IP address of the actor.
`actorLocation` (ActorLocation)	A readable representation of the actor's location.
`actorLogin` (String)	The username of the user who initiated the action.
`actorResourcePath` (URI)	The HTTP path for the actor.
`actorUrl` (URI)	The HTTP URL for the actor.
`createdAt` (PreciseDateTime!)	The time the action was initiated.
`enterpriseResourcePath` (URI)	The HTTP path for this enterprise.
`enterpriseSlug` (String)	The slug of the enterprise.
`enterpriseUrl` (URI)	The HTTP URL for this enterprise.
`operationType` (OperationType)	The corresponding operation type for the action.
`organization` (Organization)	The Organization associated with the Audit Entry.
`organizationName` (String)	The name of the Organization.
`organizationResourcePath` (URI)	The HTTP path for the organization.
`organizationUrl` (URI)	The HTTP URL for the organization.
`user` (User)	The user affected by the action.
`userLogin` (String)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (URI)	The HTTP path for the user.
`userUrl` (URI)	The HTTP URL for the user.

OrgOauthAppAccessApprovedAuditEntry

Audit log entry for a org.oauth_app_access_approved event.

`OrgOauthAppAccessApprovedAuditEntry` Implements

- AuditEntry
- Node
- OauthApplicationAuditEntryData
- OrganizationAuditEntryData

Fields for `OrgOauthAppAccessApprovedAuditEntry`

Name	Description
`action` (String!)	The action name.
`actor` (AuditEntryActor)	The user who initiated the action.
`actorIp` (String)	The IP address of the actor.
`actorLocation` (ActorLocation)	A readable representation of the actor's location.
`actorLogin` (String)	The username of the user who initiated the action.
`actorResourcePath` (URI)	The HTTP path for the actor.
`actorUrl` (URI)	The HTTP URL for the actor.
`createdAt` (PreciseDateTime!)	The time the action was initiated.
`oauthApplicationName` (String)	The name of the OAuth application.
`oauthApplicationResourcePath` (URI)	The HTTP path for the OAuth application.
`oauthApplicationUrl` (URI)	The HTTP URL for the OAuth application.
`operationType` (OperationType)	The corresponding operation type for the action.
`organization` (Organization)	The Organization associated with the Audit Entry.
`organizationName` (String)	The name of the Organization.
`organizationResourcePath` (URI)	The HTTP path for the organization.
`organizationUrl` (URI)	The HTTP URL for the organization.
`user` (User)	The user affected by the action.
`userLogin` (String)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (URI)	The HTTP path for the user.
`userUrl` (URI)	The HTTP URL for the user.

OrgOauthAppAccessBlockedAuditEntry

Audit log entry for a org.oauth_app_access_blocked event.

`OrgOauthAppAccessBlockedAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OauthApplicationAuditEntryData`
- `OrganizationAuditEntryData`

Fields for `OrgOauthAppAccessBlockedAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`oauthApplicationName` (`String`)	The name of the OAuth application.
`oauthApplicationResourcePath` (`URI`)	The HTTP path for the OAuth application.
`oauthApplicationUrl` (`URI`)	The HTTP URL for the OAuth application.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgOauthAppAccessDeniedAuditEntry

Audit log entry for a org.oauth_app_access_denied event.

`OrgOauthAppAccessDeniedAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OauthApplicationAuditEntryData`
- `OrganizationAuditEntryData`

Fields for `OrgOauthAppAccessDeniedAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`oauthApplicationName` (`String`)	The name of the OAuth application.
`oauthApplicationResourcePath` (`URI`)	The HTTP path for the OAuth application.
`oauthApplicationUrl` (`URI`)	The HTTP URL for the OAuth application.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgOauthAppAccessRequestedAuditEntry

Audit log entry for a org.oauth_app_access_requested event.

`OrgOauthAppAccessRequestedAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OauthApplicationAuditEntryData`
- `OrganizationAuditEntryData`

Fields for `OrgOauthAppAccessRequestedAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`oauthApplicationName` (`String`)	The name of the OAuth application.
`oauthApplicationResourcePath` (`URI`)	The HTTP path for the OAuth application.
`oauthApplicationUrl` (`URI`)	The HTTP URL for the OAuth application.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgOauthAppAccessUnblockedAuditEntry

Audit log entry for a org.oauth_app_access_unblocked event.

`OrgOauthAppAccessUnblockedAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OauthApplicationAuditEntryData`
- `OrganizationAuditEntryData`

Fields for `OrgOauthAppAccessUnblockedAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`oauthApplicationName` (`String`)	The name of the OAuth application.
`oauthApplicationResourcePath` (`URI`)	The HTTP path for the OAuth application.
`oauthApplicationUrl` (`URI`)	The HTTP URL for the OAuth application.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgRemoveBillingManagerAuditEntry

Audit log entry for a org.remove_billing_manager event.

`OrgRemoveBillingManagerAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgRemoveBillingManagerAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`reason` (`OrgRemoveBillingManagerAuditEntryReason`)	The reason for the billing manager being removed.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgRemoveMemberAuditEntry

Audit log entry for a org.remove_member event.

`OrgRemoveMemberAuditEntry` Implements

- `AuditEntry`

- Node
- OrganizationAuditEntryData

Fields for `OrgRemoveMemberAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`membershipTypes` (`[OrgRemoveMemberAuditEntryMembershipType!]`)	The types of membership the member has with the organization.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`reason` (`OrgRemoveMemberAuditEntryReason`)	The reason for the member being removed.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgRemoveOutsideCollaboratorAuditEntry

Audit log entry for a org.remove_outside_collaborator event.

`OrgRemoveOutsideCollaboratorAuditEntry` Implements

- AuditEntry

- Node
- OrganizationAuditEntryData

Fields for `OrgRemoveOutsideCollaboratorAuditEntry`

Name	Description
`action` (String!)	The action name.
`actor` (AuditEntryActor)	The user who initiated the action.
`actorIp` (String)	The IP address of the actor.
`actorLocation` (ActorLocation)	A readable representation of the actor's location.
`actorLogin` (String)	The username of the user who initiated the action.
`actorResourcePath` (URI)	The HTTP path for the actor.
`actorUrl` (URI)	The HTTP URL for the actor.
`createdAt` (PreciseDateTime!)	The time the action was initiated.
`membershipTypes` ([OrgRemoveOutsideCollaboratorAuditEntryMembershipType!])	The types of membership the outside collaborator has with the organization.
`operationType` (OperationType)	The corresponding operation type for the action.
`organization` (Organization)	The Organization associated with the Audit Entry.
`organizationName` (String)	The name of the Organization.
`organizationResourcePath` (URI)	The HTTP path for the organization.
`organizationUrl` (URI)	The HTTP URL for the organization.
`reason` (OrgRemoveOutsideCollaboratorAuditEntryReason)	The reason for the outside collaborator being removed from the Organization.
`user` (User)	The user affected by the action.
`userLogin` (String)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (URI)	The HTTP path for the user.
`userUrl` (URI)	The HTTP URL for the user.

OrgRestoreMemberAuditEntry

Audit log entry for a org.restore_member event.

`OrgRestoreMemberAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgRestoreMemberAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`restoredCustomEmailRoutingsCount` (`Int`)	The number of custom email routings for the restored member.
`restoredIssueAssignmentsCount` (`Int`)	The number of issue assignments for the restored member.
`restoredMemberships` (`[OrgRestoreMemberAuditEntryMembership!]`)	Restored organization membership objects.
`restoredMembershipsCount` (`Int`)	The number of restored memberships.
`restoredRepositoriesCount` (`Int`)	The number of repositories of the restored member.
`restoredRepositoryStarsCount` (`Int`)	The number of starred repositories for the restored member.
`restoredRepositoryWatchesCount` (`Int`)	The number of watched repositories for the restored member.
`user` (`User`)	The user affected by the action.

`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgRestoreMemberMembershipOrganizationAuditEntryData

Metadata for an organization membership for org.restore_member actions.

`OrgRestoreMemberMembershipOrganizationAuditEntryData` Implements

- `OrganizationAuditEntryData`

Fields for `OrgRestoreMemberMembershipOrganizationAuditEntryData`

Name	Description
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.

OrgRestoreMemberMembershipRepositoryAuditEntryData

Metadata for a repository membership for org.restore_member actions.

`OrgRestoreMemberMembershipRepositoryAuditEntryData` Implements

- `RepositoryAuditEntryData`

Fields for `OrgRestoreMemberMembershipRepositoryAuditEntryData`

Name	Description
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.

OrgRestoreMemberMembershipTeamAuditEntryData

Metadata for a team membership for org.restore_member actions.

`OrgRestoreMemberMembershipTeamAuditEntryData` Implements

- `TeamAuditEntryData`

Fields for `OrgRestoreMemberMembershipTeamAuditEntryData`

Name	Description
`team` (`Team`)	The team associated with the action.
`teamName` (`String`)	The name of the team.
`teamResourcePath` (`URI`)	The HTTP path for this team.
`teamUrl` (`URI`)	The HTTP URL for this team.

OrgUnblockUserAuditEntry

Audit log entry for a org.unblock_user.

`OrgUnblockUserAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgUnblockUserAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`blockedUser` (`User`)	The user being unblocked by the organization.
`blockedUserName` (`String`)	The username of the blocked user.
`blockedUserResourcePath` (`URI`)	The HTTP path for the blocked user.

`blockedUserUrl` (`URI`)	The HTTP URL for the blocked user.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgUpdateDefaultRepositoryPermissionAuditEntry

Audit log entry for a org.update_default_repository_permission.

`OrgUpdateDefaultRepositoryPermissionAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgUpdateDefaultRepositoryPermissionAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.

`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`permission` (`OrgUpdateDefaultRepositoryPermissionAuditEntryPermission`)	The new base repository permission level for the organization.
`permissionWas` (`OrgUpdateDefaultRepositoryPermissionAuditEntryPermission`)	The former base repository permission level for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgUpdateMemberAuditEntry

Audit log entry for a org.update_member event.

`OrgUpdateMemberAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgUpdateMemberAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.

`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`permission` (`OrgUpdateMemberAuditEntryPermission`)	The new member permission level for the organization.
`permissionWas` (`OrgUpdateMemberAuditEntryPermission`)	The former member permission level for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

OrgUpdateMemberRepositoryCreationPermissionAuditEntry

Audit log entry for a org.update_member_repository_creation_permission event.

`OrgUpdateMemberRepositoryCreationPermissionAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgUpdateMemberRepositoryCreationPermissionAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.

`actorUrl` (`URI`)	The HTTP URL for the actor.
`canCreateRepositories` (`Boolean`)	Can members create repositories in the organization.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.
`visibility` (`OrgUpdateMemberRepositoryCreationPermissionAuditEntryVisibility`)	The permission for visibility level of repositories for this organization.

OrgUpdateMemberRepositoryInvitationPermissionAuditEntry

Audit log entry for a org.update_member_repository_invitation_permission event.

`OrgUpdateMemberRepositoryInvitationPermissionAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`

Fields for `OrgUpdateMemberRepositoryInvitationPermissionAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.

`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`canInviteOutsideCollaboratorsToRepositories` (`Boolean`)	Can outside collaborators be invited to repositories in the organization.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

Organization

An account on GitHub, with one or more owners, that has repositories, members and teams.

`Organization` Implements

- `Actor`
- `AnnouncementBanner`
- `MemberStatusable`
- `Node`
- `PackageOwner`
- `ProfileOwner`
- `ProjectOwner`
- `ProjectV2Owner`
- `ProjectV2Recent`
- `RepositoryDiscussionAuthor`
- `RepositoryDiscussionCommentAuthor`
- `RepositoryOwner`
- `Sponsorable`
- `UniformResourceLocatable`

Fields for `Organization`

Name	Description
`announcement` (`String`)	The text of the announcement.
`announcementExpiresAt` (`DateTime`)	The expiration date of the announcement, if any.
`announcementUserDismissible` (`Boolean`)	Whether the announcement can be dismissed by the user.
`anyPinnableItems` (`Boolean!`)	Determine if this repository owner has any items that can be pinned to their profile. **Arguments for `anyPinnableItems`** `type` (`PinnableItemType`) Filter to only a particular kind of pinnable item.
`archivedAt` (`DateTime`)	Identifies the date and time when the organization was archived.
`auditLog` (`OrganizationAuditEntryConnection!`)	Audit log entries of the organization. **Arguments for `auditLog`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`AuditLogOrder`) Ordering options for the returned audit log entries. `query` (`String`) The query string to filter audit entries.
`avatarUrl` (`URI!`)	A URL pointing to the organization's public avatar. **Arguments for `avatarUrl`** `size` (`Int`) The size of the resulting square image.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.

created.

`databaseId` (`Int`)	Identifies the primary key from the database.
`description` (`String`)	The organization's public profile description.
`descriptionHTML` (`String`)	The organization's public profile description rendered to HTML.
`domains` (`VerifiableDomainConnection`)	A list of domains owned by the organization. **Arguments for `domains`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `isApproved` (`Boolean`) Filter by if the domain is approved. `isVerified` (`Boolean`) Filter by if the domain is verified. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`VerifiableDomainOrder`) Ordering options for verifiable domains returned.
`email` (`String`)	The organization's public email.
`enterpriseOwners` (`OrganizationEnterpriseOwnerConnection!`)	A list of owners of the organization's enterprise account. **Arguments for `enterpriseOwners`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`OrgEnterpriseOwnerOrder`) Ordering options for enterprise owners returned from the connection.

`organizationRole` (`RoleInOrganization`)
The organization role to filter by.

`query` (`String`)
The search string to look for.

`estimatedNextSponsorsPayoutInCents` (`Int!`)	The estimated next GitHub Sponsors payout for this user/organization in cents (USD).
`hasSponsorsListing` (`Boolean!`)	True if this user/organization has a GitHub Sponsors listing.
`interactionAbility` (`RepositoryInteractionAbility`)	The interaction ability settings for this organization.
`ipAllowListEnabledSetting` (`IpAllowListEnabledSettingValue!`)	The setting value for whether the organization has an IP allow list enabled.
`ipAllowListEntries` (`IpAllowListEntryConnection!`)	The IP addresses that are allowed to access resources owned by the organization.

Arguments for `ipAllowListEntries`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`IpAllowListEntryOrder`)
Ordering options for IP allow list entries returned.

`ipAllowListForInstalledAppsEnabledSetting` (`IpAllowListForInstalledAppsEnabledSettingValue!`)	The setting value for whether the organization has IP allow list configuration for installed GitHub Apps enabled.
`isSponsoredBy` (`Boolean!`)	Whether the given account is sponsoring this user/organization.

Arguments for `isSponsoredBy`

`accountLogin` (`String!`)
The target account's login.

`isSponsoringViewer` (`Boolean!`)	True if the viewer is sponsored by this user/organization.
`isVerified` (`Boolean!`)	Whether the organization has verified its profile

`isVerified` (`Boolean!`)	Whether the organization has verified its profile email and website.
`itemShowcase` (`ProfileItemShowcase!`)	Showcases a selection of repositories and gists that the profile owner has either curated or that have been selected automatically based on popularity.
`location` (`String`)	The organization's public profile location.
`login` (`String!`)	The organization's login name.
`mannequins` (`MannequinConnection!`)	A list of all mannequins for this organization. **Arguments for `mannequins`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `login` (`String`) Filter mannequins by login. `orderBy` (`MannequinOrder`) Ordering options for mannequins returned from the connection.
`memberStatuses` (`UserStatusConnection!`)	Get the status messages members of this entity have set that are either public or visible only to the organization. **Arguments for `memberStatuses`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`UserStatusOrder`) Ordering options for user statuses returned from the connection.

`membersCanForkPrivateRepositories` (`Boolean!`)	Members can fork private repositories in this organization.
`membersWithRole` (`OrganizationMemberConnection!`)	A list of users who are members of this organization. **Arguments for `membersWithRole`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`monthlyEstimatedSponsorsIncomeInCents` (`Int!`)	The estimated monthly GitHub Sponsors income for this user/organization in cents (USD).
`name` (`String`)	The organization's public profile name.
`newTeamResourcePath` (`URI!`)	The HTTP path creating a new team.
`newTeamUrl` (`URI!`)	The HTTP URL creating a new team.
`notificationDeliveryRestrictionEnabledSetting` (`NotificationRestrictionSettingValue!`)	Indicates if email notification delivery for this organization is restricted to verified or approved domains.
`organizationBillingEmail` (`String`)	The billing email for the organization.
`packages` (`PackageConnection!`)	A list of packages under the owner. **Arguments for `packages`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `names` (`[String]`) Find packages by their names. `orderBy` (`PackageOrder`) Ordering of the returned packages.

`packageType` (`PackageType`)
Filter registry package by type.

`repositoryId` (`ID`)
Find packages in a repository by ID.

`pendingMembers` (`UserConnection!`)

A list of users who have been invited to join this organization.

Arguments for `pendingMembers`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`pinnableItems` (`PinnableItemConnection!`)

A list of repositories and gists this profile owner can pin to their profile.

Arguments for `pinnableItems`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`types` (`[PinnableItemType!]`)
Filter the types of pinnable items that are returned.

`pinnedItems` (`PinnableItemConnection!`)

A list of repositories and gists this profile owner has pinned to their profile.

Arguments for `pinnedItems`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`types` (`[PinnableItemType!]`)
Filter the types of pinned items that are returned.

`pinnedItemsRemaining` (`Int!`)

Returns how many more items this profile owner can pin to their profile.

`project` (`Project`)

Find project by number.

Arguments for `project`

`number` (`Int!`)
The project number to find.

`projectV2` (`ProjectV2`)

Find a project by number.

Arguments for `projectV2`

`number` (`Int!`)
The project number.

`projects` (`ProjectConnection!`)

A list of projects under the owner.

Arguments for `projects`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ProjectOrder`)
Ordering options for projects returned from the connection.

`search` (`String`)
Query to search projects by, currently only searching by name.

`states` (`[ProjectState!]`)

`states` (`[ProjectState!]`)
A list of states to filter the projects by.

`projectsResourcePath` (`URI!`)	The HTTP path listing organization's projects.
`projectsUrl` (`URI!`)	The HTTP URL listing organization's projects.
`projectsV2` (`ProjectV2Connection!`)	A list of projects under the owner.

Arguments for `projectsV2`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ProjectV2Order`)
How to order the returned projects.

`query` (`String`)
A project to search for under the the owner.

`recentProjects` (`ProjectV2Connection!`)	Recent projects that this user has modified in the context of the owner.

Arguments for `recentProjects`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`repositories` (`RepositoryConnection!`)	A list of repositories that the user owns.

Arguments for `repositories`

`affiliations` (`[RepositoryAffiliation]`)
Array of viewer's affiliation options for repositories returned from the connection. For example, OWNER will include only

For example, OWNER will include only repositories that the current viewer owns.

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`hasIssuesEnabled` (`Boolean`)
If non-null, filters repositories according to whether they have issues enabled.

`isArchived` (`Boolean`)
If non-null, filters repositories according to whether they are archived and not maintained.

`isFork` (`Boolean`)
If non-null, filters repositories according to whether they are forks of another repository.

`isLocked` (`Boolean`)
If non-null, filters repositories according to whether they have been locked.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`RepositoryOrder`)
Ordering options for repositories returned from the connection.

`ownerAffiliations` (`[RepositoryAffiliation]`)
Array of owner's affiliation options for repositories returned from the connection. For example, OWNER will include only repositories that the organization or user being viewed owns.

`privacy` (`RepositoryPrivacy`)
If non-null, filters repositories according to privacy.

`repository` (`Repository`)

Find Repository.

Arguments for `repository`

`followRenames` (`Boolean`)
Follow repository renames. If disabled, a repository referenced by its old name will return an error.
The default value is `true`.

`name` (`String!`)

Name of Repository to find.

`repositoryDiscussionComments` (`DiscussionCommentConnection!`)

Discussion comments this user has authored.

Arguments for `repositoryDiscussionComments`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`onlyAnswers` (`Boolean`)
Filter discussion comments to only those that were marked as the answer.
The default value is `false`.

`repositoryId` (`ID`)
Filter discussion comments to only those in a specific repository.

`repositoryDiscussions` (`DiscussionConnection!`)

Discussions this user has started.

Arguments for `repositoryDiscussions`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`answered` (`Boolean`)
Filter discussions to only those that have been answered or not. Defaults to including both answered and unanswered discussions.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`DiscussionOrder`)
Ordering options for discussions returned from the connection.

`repositoryId` (`ID`)
Filter discussions to only those in a specific

repository.

`states` (`[DiscussionState!]`)
A list of states to filter the discussions by.

`repositoryMigrations` (`RepositoryMigrationConnection!`)

A list of all repository migrations for this organization.

Arguments for `repositoryMigrations`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`RepositoryMigrationOrder`)
Ordering options for repository migrations returned.

`repositoryName` (`String`)
Filter repository migrations by repository name.

`state` (`MigrationState`)
Filter repository migrations by state.

`requiresTwoFactorAuthentication` (`Boolean`)

When true the organization requires all members, billing managers, and outside collaborators to enable two-factor authentication.

`resourcePath` (`URI!`)

The HTTP path for this organization.

`ruleset` (`RepositoryRuleset`)

Returns a single ruleset from the current organization by ID.

Arguments for `ruleset`

`databaseId` (`Int!`)
The ID of the ruleset to be returned.

`rulesets` (`RepositoryRulesetConnection`)

A list of rulesets for this organization.

Arguments for `rulesets`

`after` (`String`)
Returns the elements in the list that come

after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`includeParents` (`Boolean`)
Return rulesets configured at higher levels that apply to this organization.
The default value is `true`.

`last` (`Int`)
Returns the last *n* elements from the list.

`samlIdentityProvider` (`OrganizationIdentityProvider`)

The Organization's SAML identity provider. Visible to (1) organization owners, (2) organization owners' personal access tokens (classic) with read:org or admin:org scope, (3) GitHub App with an installation token with read or write access to members.

`sponsoring` (`SponsorConnection!`)

List of users and organizations this entity is sponsoring.

Arguments for `sponsoring`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`SponsorOrder`)
Ordering options for the users and organizations returned from the connection.

`sponsors` (`SponsorConnection!`)

List of sponsors for this user or organization.

Arguments for `sponsors`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`SponsorOrder`)
Ordering options for sponsors returned from the connection.

`tierId` (`ID`)
If given, will filter for sponsors at the given tier. Will only return sponsors whose tier the viewer is permitted to see.

`sponsorsActivities` (`SponsorsActivityConnection!`)

Events involving this sponsorable, such as new sponsorships.

Arguments for `sponsorsActivities`

`actions` (`[SponsorsActivityAction!]`)
Filter activities to only the specified actions.

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`includeAsSponsor` (`Boolean`)
Whether to include those events where this sponsorable acted as the sponsor. Defaults to only including events where this sponsorable was the recipient of a sponsorship.
The default value is `false`.

`includePrivate` (`Boolean`)
Whether or not to include private activities in the result set. Defaults to including public and private activities.
The default value is `true`.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`SponsorsActivityOrder`)
Ordering options for activity returned from the connection.

`period` (`SponsorsActivityPeriod`)
Filter activities returned to only those that occurred in the most recent specified time period. Set to ALL to avoid filtering by when the activity occurred. Will be ignored if `since` or `until` is given.
The default value is `MONTH`.

`since` (`DateTime`)
Filter activities to those that occurred on or after this time.

`until` (`DateTime`)
Filter activities to those that occurred before this time.

`sponsorsListing` (`SponsorsListing`)	The GitHub Sponsors listing for this user or organization.
`sponsorshipForViewerAsSponsor` (`Sponsorship`)	The sponsorship from the viewer to this user/organization; that is, the sponsorship where you're the sponsor.

Arguments for `sponsorshipForViewerAsSponsor`

`activeOnly` (`Boolean`)
Whether to return the sponsorship only if it's still active. Pass false to get the viewer's sponsorship back even if it has been cancelled.
The default value is `true`.

`sponsorshipForViewerAsSponsorable` (`Sponsorship`)	The sponsorship from this user/organization to the viewer; that is, the sponsorship you're receiving.

Arguments for `sponsorshipForViewerAsSponsorable`

`activeOnly` (`Boolean`)
Whether to return the sponsorship only if it's still active. Pass false to get the sponsorship back even if it has been cancelled.
The default value is `true`.

`sponsorshipNewsletters` (`SponsorshipNewsletterConnection!`)	List of sponsorship updates sent from this sponsorable to sponsors.

Arguments for `sponsorshipNewsletters`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`SponsorshipNewsletterOrder`)
Ordering options for sponsorship updates returned from the connection.

`sponsorshipsAsMaintainer` (`SponsorshipConnection!`)

The sponsorships where this user or organization is the maintainer receiving the funds.

Arguments for `sponsorshipsAsMaintainer`

`activeOnly` (`Boolean`)
Whether to include only sponsorships that are active right now, versus all sponsorships this maintainer has ever received.
The default value is `true`.

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`includePrivate` (`Boolean`)
Whether or not to include private sponsorships in the result set.
The default value is `false`.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`SponsorshipOrder`)
Ordering options for sponsorships returned from this connection. If left blank, the sponsorships will be ordered based on relevancy to the viewer.

`sponsorshipsAsSponsor` (`SponsorshipConnection!`)

The sponsorships where this user or organization is the funder.

Arguments for `sponsorshipsAsSponsor`

`activeOnly` (`Boolean`)
Whether to include only sponsorships that are active right now, versus all sponsorships this sponsor has ever made.
The default value is `true`.

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come

before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`maintainerLogins` (`[String!]`)
Filter sponsorships returned to those for the specified maintainers. That is, the recipient of the sponsorship is a user or organization with one of the given logins.

`orderBy` (`SponsorshipOrder`)
Ordering options for sponsorships returned from this connection. If left blank, the sponsorships will be ordered based on relevancy to the viewer.

`team` (`Team`)

Find an organization's team by its slug.

Arguments for `team`

`slug` (`String!`)
The name or slug of the team to find.

`teams` (`TeamConnection!`)

A list of teams in this organization.

Arguments for `teams`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`ldapMapped` (`Boolean`)
If true, filters teams that are mapped to an LDAP Group (Enterprise only).

`notificationSetting` (`TeamNotificationSetting`)
If non-null, filters teams according to notification setting.

`orderBy` (`TeamOrder`)
Ordering options for teams returned from the connection.

`privacy` (`TeamPrivacy`)
If non-null, filters teams according to privacy.

`query` (`String`)
If non-null, filters teams with query on team name and team slug.

`role` (`TeamRole`)
If non-null, filters teams according to whether the viewer is an admin or member on team.

`rootTeamsOnly` (`Boolean`)
If true, restrict to only root teams.
The default value is `false`.

`userLogins` (`[String!]`)
User logins to filter by.

`teamsResourcePath` (`URI!`)	The HTTP path listing organization's teams.
`teamsUrl` (`URI!`)	The HTTP URL listing organization's teams.
`totalSponsorshipAmountAsSponsorInCents` (`Int`)	The amount in United States cents (e.g., 500 = $5.00 USD) that this entity has spent on GitHub to fund sponsorships. Only returns a value when viewed by the user themselves or by a user who can manage sponsorships for the requested organization.

Arguments for `totalSponsorshipAmountAsSponsorInCents`

`since` (`DateTime`)
Filter payments to those that occurred on or after this time.

`sponsorableLogins` (`[String!]`)
Filter payments to those made to the users or organizations with the specified usernames.

`until` (`DateTime`)
Filter payments to those that occurred before this time.

`twitterUsername` (`String`)	The organization's Twitter username.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this organization.
`viewerCanAdminister` (`Boolean!`)	Organization is adminable by the viewer.
`viewerCanChangePinnedItems` (`Boolean!`)	Can the viewer pin repositories and gists to the profile?.

`viewerCanCreateProjects` (`Boolean!`)	Can the current viewer create new projects on this owner.
`viewerCanCreateRepositories` (`Boolean!`)	Viewer can create repositories on this organization.
`viewerCanCreateTeams` (`Boolean!`)	Viewer can create teams on this organization.
`viewerCanSponsor` (`Boolean!`)	Whether or not the viewer is able to sponsor this user/organization.
`viewerIsAMember` (`Boolean!`)	Viewer is an active member of this organization.
`viewerIsFollowing` (`Boolean!`)	Whether or not this Organization is followed by the viewer.
`viewerIsSponsoring` (`Boolean!`)	True if the viewer is sponsoring this user/organization.
`webCommitSignoffRequired` (`Boolean!`)	Whether contributors are required to sign off on web-based commits for repositories in this organization.
`websiteUrl` (`URI`)	The organization's public profile URL.

OrganizationAuditEntryConnection

The connection type for OrganizationAuditEntry.

Fields for `OrganizationAuditEntryConnection`

Name	Description
`edges` (`[OrganizationAuditEntryEdge]`)	A list of edges.
`nodes` (`[OrganizationAuditEntry]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

OrganizationAuditEntryEdge

An edge in a connection.

Fields for `OrganizationAuditEntryEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`OrganizationAuditEntry`)	The item at the end of the edge.

OrganizationConnection

A list of organizations managed by an enterprise.

Fields for `OrganizationConnection`

Name	Description
`edges` (`[OrganizationEdge]`)	A list of edges.
`nodes` (`[Organization]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

OrganizationEdge

An edge in a connection.

Fields for `OrganizationEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Organization`)	The item at the end of the edge.

OrganizationEnterpriseOwnerConnection

The connection type for User.

Fields for `OrganizationEnterpriseOwnerConnection`

Name	Description
`edges` (`[OrganizationEnterpriseOwnerEdge]`)	A list of edges.
`nodes` (`[User]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

OrganizationEnterpriseOwnerEdge

An enterprise owner in the context of an organization that is part of the enterprise.

Fields for `OrganizationEnterpriseOwnerEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.

`node` (`User`)	The item at the end of the edge.
`organizationRole` (`RoleInOrganization!`)	The role of the owner with respect to the organization.

OrganizationIdentityProvider

An Identity Provider configured to provision SAML and SCIM identities for Organizations. Visible to (1) organization owners, (2) organization owners' personal access tokens (classic) with read:org or admin:org scope, (3) GitHub App with an installation token with read or write access to members.

`OrganizationIdentityProvider` Implements

- `Node`

Fields for `OrganizationIdentityProvider`

Name	Description
`digestMethod` (`URI`)	The digest algorithm used to sign SAML requests for the Identity Provider.
`externalIdentities` (`ExternalIdentityConnection!`)	External Identities provisioned by this Identity Provider. **Arguments for `externalIdentities`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `login` (`String`) Filter to external identities with the users login. `membersOnly` (`Boolean`) Filter to external identities with valid org membership only. `userName` (`String`) Filter to external identities with the users userName/NameID attribute.
`idpCertificate` (`X509Certificate`)	The x509 certificate used by the Identity Provider to sign assertions and responses.

`issuer` (`String`)	The Issuer Entity ID for the SAML Identity Provider.
`organization` (`Organization`)	Organization this Identity Provider belongs to.
`signatureMethod` (`URI`)	The signature algorithm used to sign SAML requests for the Identity Provider.
`ssoUrl` (`URI`)	The URL endpoint for the Identity Provider's SAML SSO.

OrganizationInvitation

An Invitation for a user to an organization.

`OrganizationInvitation` Implements

- `Node`

Fields for `OrganizationInvitation`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`email` (`String`)	The email address of the user invited to the organization.
`invitationSource` (`OrganizationInvitationSource!`)	The source of the invitation.
`invitationType` (`OrganizationInvitationType!`)	The type of invitation that was sent (e.g. email, user).
`invitee` (`User`)	The user who was invited to the organization.
`inviter` (`User!`)	The user who created the invitation.
`organization` (`Organization!`)	The organization the invite is for.
`role` (`OrganizationInvitationRole!`)	The user's pending role in the organization (e.g. member, owner).

OrganizationInvitationConnection

The connection type for OrganizationInvitation.

Fields for `OrganizationInvitationConnection`

Name	Description
`edges` (`[OrganizationInvitationEdge]`)	A list of edges.
`nodes` (`[OrganizationInvitation]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.

`totalCount` (`Int!`)	Identifies the total count of items in the connection.

OrganizationInvitationEdge

An edge in a connection.

Fields for `OrganizationInvitationEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`OrganizationInvitation`)	The item at the end of the edge.

OrganizationMemberConnection

The connection type for User.

Fields for `OrganizationMemberConnection`

Name	Description
`edges` (`[OrganizationMemberEdge]`)	A list of edges.
`nodes` (`[User]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

OrganizationMemberEdge

Represents a user within an organization.

Fields for `OrganizationMemberEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`hasTwoFactorEnabled` (`Boolean`)	Whether the organization member has two factor enabled or not. Returns null if information is not available to viewer.
`node` (`User`)	The item at the end of the edge.
`role` (`OrganizationMemberRole`)	The role this user has in the organization.

OrganizationMigration

A GitHub Enterprise Importer (GEI) organization migration.

`OrganizationMigration` Implements

- `Node`

Fields for `OrganizationMigration`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`String`)	Identifies the primary key from the database.
`failureReason` (`String`)	The reason the organization migration failed.
`remainingRepositoriesCount` (`Int`)	The remaining amount of repos to be migrated.
`sourceOrgName` (`String!`)	The name of the source organization to be migrated.
`sourceOrgUrl` (`URI!`)	The URL of the source organization to migrate.
`state` (`OrganizationMigrationState!`)	The migration state.
`targetOrgName` (`String!`)	The name of the target organization.
`totalRepositoriesCount` (`Int`)	The total amount of repositories to be migrated.

OrganizationTeamsHovercardContext

An organization teams hovercard context.

`OrganizationTeamsHovercardContext` Implements

- `HovercardContext`

Fields for `OrganizationTeamsHovercardContext`

Name	Description
`message` (`String!`)	A string describing this context.
`octicon` (`String!`)	An octicon to accompany this context.
`relevantTeams` (`TeamConnection!`)	Teams in this organization the user is a member of that are relevant. **Arguments for `relevantTeams`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`teamsResourcePath` (`URI!`)	The path for the full team list for this user.
`teamsUrl` (`URI!`)	The URL for the full team list for this user.
`totalTeamCount` (`Int!`)	The total number of teams the user is on in the organization.

OrganizationsHovercardContext

An organization list hovercard context.

`OrganizationsHovercardContext` Implements

- `HovercardContext`

Fields for `OrganizationsHovercardContext`

Name	Description
`message` (`String!`)	A string describing this context.
`octicon` (`String!`)	An octicon to accompany this context.
`relevantOrganizations` (`OrganizationConnection!`)	Organizations this user is a member of that are relevant. **Arguments for `relevantOrganizations`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`OrganizationOrder`) Ordering options for the User's organizations.
`totalOrganizationCount` (`Int!`)	The total number of organizations this user is in.

Package

Information for an uploaded package.

`Package` Implements

- `Node`

Fields for `Package`

Name	Description
`latestVersion` (`PackageVersion`)	Find the latest version for the package.
`name` (`String!`)	Identifies the name of the package.
`packageType` (`PackageType!`)	Identifies the type of the package.
`repository` (`Repository`)	The repository this package belongs to.
`statistics` (`PackageStatistics`)	Statistics about package activity.
`version` (`PackageVersion`)	Find package version by version string. **Arguments for `version`** `version` (`String!`) The package version.
`versions` (`PackageVersionConnection!`)	list of versions for this package. **Arguments for `versions`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`PackageVersionOrder`) Ordering of the returned packages.

PackageConnection

The connection type for Package.

Fields for `PackageConnection`

Name	Description
`edges` (`[PackageEdge]`)	A list of edges.
`nodes` (`[Package]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

PackageEdge

An edge in a connection.

Fields for `PackageEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Package`)	The item at the end of the edge.

PackageFile

A file in a package version.

`PackageFile` Implements

- `Node`

Fields for `PackageFile`

Name	Description
`md5` (`String`)	MD5 hash of the file.
`name` (`String!`)	Name of the file.
`packageVersion` (`PackageVersion`)	The package version this file belongs to.
`sha1` (`String`)	SHA1 hash of the file.
`sha256` (`String`)	SHA256 hash of the file.
`size` (`Int`)	Size of the file in bytes.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI`)	URL to download the asset.

PackageFileConnection

The connection type for PackageFile.

Fields for `PackageFileConnection`

Name	Description
`edges` (`[PackageFileEdge]`)	A list of edges.
`nodes` (`[PackageFile]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

PackageFileEdge

An edge in a connection.

Fields for `PackageFileEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`PackageFile`)	The item at the end of the edge.

PackageStatistics

Represents a object that contains package activity statistics such as downloads.

Fields for `PackageStatistics`

Name	Description
`downloadsTotalCount` (`Int!`)	Number of times the package was downloaded since it was created.

PackageTag

A version tag contains the mapping between a tag name and a version.

`PackageTag` Implements

- `Node`

Fields for `PackageTag`

Name	Description
`name` (`String!`)	Identifies the tag name of the version.
`version` (`PackageVersion`)	Version that the tag is associated with.

PackageVersion

Information about a specific package version.

`PackageVersion` Implements

- `Node`

Fields for `PackageVersion`

Name	Description
`files` (`PackageFileConnection!`)	List of files associated with this package version. **Arguments for `files`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`PackageFileOrder`) Ordering of the returned package files.
`package` (`Package`)	The package associated with this version.
`platform` (`String`)	The platform this version was built for.
`preRelease` (`Boolean!`)	Whether or not this version is a pre-release.
`readme` (`String`)	The README of this package version.
`release` (`Release`)	The release associated with this package version.
`statistics` (`PackageVersionStatistics`)	Statistics about package activity.
`summary` (`String`)	The package version summary.
`version` (`String!`)	The version string.

PackageVersionConnection

The connection type for PackageVersion.

Fields for `PackageVersionConnection`

Name	Description
`edges` (`[PackageVersionEdge]`)	A list of edges.
`nodes` (`[PackageVersion]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

PackageVersionEdge

An edge in a connection.

Fields for `PackageVersionEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`PackageVersion`)	The item at the end of the edge.

PackageVersionStatistics

Represents a object that contains package version activity statistics such as downloads.

Fields for `PackageVersionStatistics`

Name	Description
`downloadsTotalCount` (`Int!`)	Number of times the package was downloaded since it was created.

PageInfo

Information about pagination in a connection.

Fields for `PageInfo`

Name	Description
`endCursor` (`String`)	When paginating forwards, the cursor to continue.
`hasNextPage` (`Boolean!`)	When paginating forwards, are there more items?.

`hasPreviousPage` (`Boolean!`)	When paginating backwards, are there more items?.
`startCursor` (`String`)	When paginating backwards, the cursor to continue.

PermissionSource

A level of permission and source for a user's access to a repository.

Fields for `PermissionSource`

Name	Description
`organization` (`Organization!`)	The organization the repository belongs to.
`permission` (`DefaultRepositoryPermissionField!`)	The level of access this source has granted to the user.
`roleName` (`String`)	The name of the role this source has granted to the user.
`source` (`PermissionGranter!`)	The source of this permission.

PinnableItemConnection

The connection type for PinnableItem.

Fields for `PinnableItemConnection`

Name	Description
`edges` (`[PinnableItemEdge]`)	A list of edges.
`nodes` (`[PinnableItem]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

PinnableItemEdge

An edge in a connection.

Fields for `PinnableItemEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`PinnableItem`)	The item at the end of the edge.

PinnedDiscussion

A Pinned Discussion is a discussion pinned to a repository's index page.

`PinnedDiscussion` Implements

- `Node`
- `RepositoryNode`

Fields for `PinnedDiscussion`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`discussion` (`Discussion!`)	The discussion that was pinned.
`gradientStopColors` (`[String!]!`)	Color stops of the chosen gradient.
`pattern` (`PinnedDiscussionPattern!`)	Background texture pattern.
`pinnedBy` (`Actor!`)	The actor that pinned this discussion.
`preconfiguredGradient` (`PinnedDiscussionGradient`)	Preconfigured background gradient option.
`repository` (`Repository!`)	The repository associated with this node.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

PinnedDiscussionConnection

The connection type for PinnedDiscussion.

Fields for `PinnedDiscussionConnection`

Name	Description
`edges` (`[PinnedDiscussionEdge]`)	A list of edges.
`nodes` (`[PinnedDiscussion]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

PinnedDiscussionEdge

An edge in a connection.

Fields for `PinnedDiscussionEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`PinnedDiscussion`)	The item at the end of the edge.

PinnedEvent

Represents a `pinned` event on a given issue or pull request.

`PinnedEvent` Implements

- `Node`

Fields for `PinnedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`issue` (`Issue!`)	Identifies the issue associated with the event.

PinnedIssue

A Pinned Issue is a issue pinned to a repository's index page.

`PinnedIssue` Implements

- `Node`

Fields for `PinnedIssue`

Name	Description
`databaseId` (`Int`)	Identifies the primary key from the database.
`fullDatabaseId` (`BigInt`)	Identifies the primary key from the database as a BigInt.
`issue` (`Issue!`)	The issue that was pinned.
`pinnedBy` (`Actor!`)	The actor that pinned this issue.
`repository` (`Repository!`)	The repository that this issue was pinned to.

PinnedIssueConnection

The connection type for PinnedIssue.

Fields for `PinnedIssueConnection`

Name	Description
`edges` (`[PinnedIssueEdge]`)	A list of edges.
`nodes` (`[PinnedIssue]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

PinnedIssueEdge

An edge in a connection.

Fields for `PinnedIssueEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`PinnedIssue`)	The item at the end of the edge.

PrivateRepositoryForkingDisableAuditEntry

Audit log entry for a private_repository_forking.disable event.

`PrivateRepositoryForkingDisableAuditEntry` Implements

- `AuditEntry`
- `EnterpriseAuditEntryData`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `PrivateRepositoryForkingDisableAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.

`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`enterpriseResourcePath` (`URI`)	The HTTP path for this enterprise.
`enterpriseSlug` (`String`)	The slug of the enterprise.
`enterpriseUrl` (`URI`)	The HTTP URL for this enterprise.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

PrivateRepositoryForkingEnableAuditEntry

Audit log entry for a private_repository_forking.enable event.

`PrivateRepositoryForkingEnableAuditEntry` Implements

- `AuditEntry`
- `EnterpriseAuditEntryData`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `PrivateRepositoryForkingEnableAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.

`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`enterpriseResourcePath` (`URI`)	The HTTP path for this enterprise.
`enterpriseSlug` (`String`)	The slug of the enterprise.
`enterpriseUrl` (`URI`)	The HTTP URL for this enterprise.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

ProfileItemShowcase

A curatable list of repositories relating to a repository owner, which defaults to showing the most popular repositories they own.

Fields for `ProfileItemShowcase`

Name	Description
`hasPinnedItems` (`Boolean!`)	Whether or not the owner has pinned any repositories or gists.

`items` (`PinnableItemConnection!`)	The repositories and gists in the showcase. If the profile owner has any pinned items, those will be returned. Otherwise, the profile owner's popular repositories will be returned. **Arguments for `items`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.

Project

Projects manage issues, pull requests and notes within a project owner.

`Project` Implements

- `Closable`
- `Node`
- `Updatable`

Fields for `Project`

Name	Description
`body` (`String`)	The project's description body.
`bodyHTML` (`HTML!`)	The projects description body rendered to HTML.
`closed` (`Boolean!`)	Indicates if the object is closed (definition of closed may depend on type).
`closedAt` (`DateTime`)	Identifies the date and time when the object was closed.
`columns` (`ProjectColumnConnection!`)	List of columns in the project. **Arguments for `columns`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor.

	`first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`creator` (`Actor`)	The actor who originally created the project.
`databaseId` (`Int`)	Identifies the primary key from the database.
`name` (`String!`)	The project's name.
`number` (`Int!`)	The project's number.
`owner` (`ProjectOwner!`)	The project's owner. Currently limited to repositories, organizations, and users.
`pendingCards` (`ProjectCardConnection!`)	List of pending cards in this project. **Arguments for `pendingCards`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `archivedStates` (`[ProjectCardArchivedState]`) A list of archived states to filter the cards by. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`progress` (`ProjectProgress!`)	Project progress details.
`resourcePath` (`URI!`)	The HTTP path for this project.
`state` (`ProjectState!`)	Whether the project is open or closed.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this project.
`viewerCanClose` (`Boolean!`)	Indicates if the object can be closed by the viewer.
`viewerCanReopen` (`Boolean!`)	Indicates if the object can be reopened by the viewer.

	viewer.
`viewerCanUpdate` (`Boolean!`)	Check if the current viewer can update this object.

ProjectCard

A card in a project.

`ProjectCard` Implements

- `Node`

Fields for `ProjectCard`

Name	Description
`column` (`ProjectColumn`)	The project column this card is associated under. A card may only belong to one project column at a time. The column field will be null if the card is created in a pending state and has yet to be associated with a column. Once cards are associated with a column, they will not become pending in the future.
`content` (`ProjectCardItem`)	The card content item.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`creator` (`Actor`)	The actor who created this card.
`databaseId` (`Int`)	Identifies the primary key from the database.
`isArchived` (`Boolean!`)	Whether the card is archived.
`note` (`String`)	The card note.
`project` (`Project!`)	The project that contains this card.
`resourcePath` (`URI!`)	The HTTP path for this card.
`state` (`ProjectCardState`)	The state of ProjectCard.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this card.

ProjectCardConnection

The connection type for ProjectCard.

Fields for `ProjectCardConnection`

Name	Description
`edges` (`[ProjectCardEdge]`)	A list of edges.

`edges` (`[ProjectCardEdge]`)	A list of edges.
`nodes` (`[ProjectCard]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ProjectCardEdge

An edge in a connection.

Fields for `ProjectCardEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`ProjectCard`)	The item at the end of the edge.

ProjectColumn

A column inside a project.

`ProjectColumn` Implements

- `Node`

Fields for `ProjectColumn`

Name	Description
`cards` (`ProjectCardConnection!`)	List of cards in the column. **Arguments for `cards`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `archivedStates` (`[ProjectCardArchivedState]`) A list of archived states to filter the cards by. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.

`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`name` (`String!`)	The project column's name.
`project` (`Project!`)	The project that contains this column.
`purpose` (`ProjectColumnPurpose`)	The semantic purpose of the column.
`resourcePath` (`URI!`)	The HTTP path for this project column.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this project column.

ProjectColumnConnection

The connection type for ProjectColumn.

Fields for `ProjectColumnConnection`

Name	Description
`edges` (`[ProjectColumnEdge]`)	A list of edges.
`nodes` (`[ProjectColumn]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ProjectColumnEdge

An edge in a connection.

Fields for `ProjectColumnEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`ProjectColumn`)	The item at the end of the edge.

ProjectConnection

A list of projects associated with the owner.

Fields for `ProjectConnection`

Name	Description
`edges` (`[ProjectEdge]`)	A list of edges.
`nodes` (`[Project]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ProjectEdge

An edge in a connection.

Fields for `ProjectEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Project`)	The item at the end of the edge.

ProjectProgress

Project progress stats.

Fields for `ProjectProgress`

Name	Description
`doneCount` (`Int!`)	The number of done cards.
`donePercentage` (`Float!`)	The percentage of done cards.
`enabled` (`Boolean!`)	Whether progress tracking is enabled and cards with purpose exist for this project.
`inProgressCount` (`Int!`)	The number of in-progress cards.
`inProgressPercentage` (`Float!`)	The percentage of in-progress cards.
`todoCount` (`Int!`)	The number of to do cards.
`todoPercentage` (`Float!`)	The percentage of to do cards.

ProjectV2

New projects that manage issues, pull requests and drafts using tables and boards.

`ProjectV2` Implements

- `Closable`
- `Node`

- `Updatable`

Fields for `ProjectV2`

Name	Description
`closed` (`Boolean!`)	Returns true if the project is closed.
`closedAt` (`DateTime`)	Identifies the date and time when the object was closed.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`creator` (`Actor`)	The actor who originally created the project.
`databaseId` (`Int`)	Identifies the primary key from the database.
`field` (`ProjectV2FieldConfiguration`)	A field of the project. **Arguments for `field`** `name` (`String!`) The name of the field.
`fields` (`ProjectV2FieldConfigurationConnection!`)	List of fields and their constraints in the project. **Arguments for `fields`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`ProjectV2FieldOrder`) Ordering options for project v2 fields returned from the connection.
`items` (`ProjectV2ItemConnection!`)	List of items in the project. **Arguments for `items`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ProjectV2ItemOrder`)
Ordering options for project v2 items returned from the connection.

`number` (`Int!`)	The project's number.
`owner` (`ProjectV2Owner!`)	The project's owner. Currently limited to organizations and users.
`public` (`Boolean!`)	Returns true if the project is public.
`readme` (`String`)	The project's readme.
`repositories` (`RepositoryConnection!`)	The repositories the project is linked to.

Arguments for `repositories`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`RepositoryOrder`)
Ordering options for repositories returned from the connection.

`resourcePath` (`URI!`)	The HTTP path for this project.
`shortDescription` (`String`)	The project's short description.
`teams` (`TeamConnection!`)	The teams the project is linked to.

Arguments for `teams`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`TeamOrder`)
Ordering options for teams returned from this connection.

`template` (`Boolean!`)	Returns true if this project is a template.
`title` (`String!`)	The project's name.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this project.
`view` (`ProjectV2View`)	A view of the project.

Arguments for `view`

`number` (`Int!`)
The number of a view belonging to the project.

`viewerCanClose` (`Boolean!`)	Indicates if the object can be closed by the viewer.
`viewerCanReopen` (`Boolean!`)	Indicates if the object can be reopened by the viewer.
`viewerCanUpdate` (`Boolean!`)	Check if the current viewer can update this object.
`views` (`ProjectV2ViewConnection!`)	List of views in the project.

Arguments for `views`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ProjectV2ViewOrder`)
Ordering options for project v2 views returned from the connection.

`workflow` (`ProjectV2Workflow`)	A workflow of the project. **Arguments for `workflow`** `number` (`Int!`) The number of a workflow belonging to the project.
`workflows` (`ProjectV2WorkflowConnection!`)	List of the workflows in the project. **Arguments for `workflows`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`ProjectV2WorkflowOrder`) Ordering options for project v2 workflows returned from the connection.

ProjectV2ActorConnection

The connection type for ProjectV2Actor.

Fields for `ProjectV2ActorConnection`

Name	Description
`edges` (`[ProjectV2ActorEdge]`)	A list of edges.
`nodes` (`[ProjectV2Actor]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ProjectV2ActorEdge

An edge in a connection.

Fields for `ProjectV2ActorEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`ProjectV2Actor`)	The item at the end of the edge.

ProjectV2Connection

The connection type for ProjectV2.

Fields for `ProjectV2Connection`

Name	Description
`edges` (`[ProjectV2Edge]`)	A list of edges.
`nodes` (`[ProjectV2]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ProjectV2Edge

An edge in a connection.

Fields for `ProjectV2Edge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`ProjectV2`)	The item at the end of the edge.

ProjectV2Field

A field inside a project.

`ProjectV2Field` Implements

- `Node`
- `ProjectV2FieldCommon`

Fields for `ProjectV2Field`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.

`dataType` (`ProjectV2FieldType!`)	The field's type.
`databaseId` (`Int`)	Identifies the primary key from the database.
`name` (`String!`)	The project field's name.
`project` (`ProjectV2!`)	The project that contains this field.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

ProjectV2FieldConfigurationConnection

The connection type for ProjectV2FieldConfiguration.

Fields for `ProjectV2FieldConfigurationConnection`

Name	Description
`edges` (`[ProjectV2FieldConfigurationEdge]`)	A list of edges.
`nodes` (`[ProjectV2FieldConfiguration]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ProjectV2FieldConfigurationEdge

An edge in a connection.

Fields for `ProjectV2FieldConfigurationEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`ProjectV2FieldConfiguration`)	The item at the end of the edge.

ProjectV2FieldConnection

The connection type for ProjectV2Field.

Fields for `ProjectV2FieldConnection`

Name	Description
`edges` (`[ProjectV2FieldEdge]`)	A list of edges.
`nodes` (`[ProjectV2Field]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the

`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ProjectV2FieldEdge

An edge in a connection.

Fields for `ProjectV2FieldEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`ProjectV2Field`)	The item at the end of the edge.

ProjectV2Item

An item within a Project.

`ProjectV2Item` Implements

- `Node`

Fields for `ProjectV2Item`

Name	Description
`content` (`ProjectV2ItemContent`)	The content of the referenced draft issue, issue, or pull request.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`creator` (`Actor`)	The actor who created the item.
`databaseId` (`Int`)	Identifies the primary key from the database.
`fieldValueByName` (`ProjectV2ItemFieldValue`)	The field value of the first project field which matches the `name` argument that is set on the item. **Arguments for `fieldValueByName`** `name` (`String!`) The name of the field to return the field value of.
`fieldValues` (`ProjectV2ItemFieldValueConnection!`)	The field values that are set on the item. **Arguments for `fieldValues`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`)

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ProjectV2ItemFieldValueOrder`)
Ordering options for project v2 item field values returned from the connection.

`isArchived` (`Boolean!`)	Whether the item is archived.
`project` (`ProjectV2!`)	The project that contains this item.
`type` (`ProjectV2ItemType!`)	The type of the item.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

ProjectV2ItemConnection

The connection type for ProjectV2Item.

Fields for `ProjectV2ItemConnection`

Name	Description
`edges` (`[ProjectV2ItemEdge]`)	A list of edges.
`nodes` (`[ProjectV2Item]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ProjectV2ItemEdge

An edge in a connection.

Fields for `ProjectV2ItemEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`ProjectV2Item`)	The item at the end of the edge.

ProjectV2ItemFieldDateValue

The value of a date field in a Project item.

`ProjectV2ItemFieldDateValue` Implements

- `Node`
- `ProjectV2ItemFieldValueCommon`

Fields for `ProjectV2ItemFieldDateValue`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`creator` (`Actor`)	The actor who created the item.
`databaseId` (`Int`)	Identifies the primary key from the database.
`date` (`Date`)	Date value for the field.
`field` (`ProjectV2FieldConfiguration!`)	The project field that contains this value.
`item` (`ProjectV2Item!`)	The project item that contains this value.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

ProjectV2ItemFieldIterationValue

The value of an iteration field in a Project item.

`ProjectV2ItemFieldIterationValue` Implements

- `Node`
- `ProjectV2ItemFieldValueCommon`

Fields for `ProjectV2ItemFieldIterationValue`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`creator` (`Actor`)	The actor who created the item.
`databaseId` (`Int`)	Identifies the primary key from the database.
`duration` (`Int!`)	The duration of the iteration in days.
`field` (`ProjectV2FieldConfiguration!`)	The project field that contains this value.
`item` (`ProjectV2Item!`)	The project item that contains this value.
`iterationId` (`String!`)	The ID of the iteration.
`startDate` (`Date!`)	The start date of the iteration.

`title` (`String!`)	The title of the iteration.
`titleHTML` (`String!`)	The title of the iteration, with HTML.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

ProjectV2ItemFieldLabelValue

The value of the labels field in a Project item.

Fields for `ProjectV2ItemFieldLabelValue`

Name	Description
`field` (`ProjectV2FieldConfiguration!`)	The field that contains this value.
`labels` (`LabelConnection`)	Labels value of a field. **Arguments for `labels`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.

ProjectV2ItemFieldMilestoneValue

The value of a milestone field in a Project item.

Fields for `ProjectV2ItemFieldMilestoneValue`

Name	Description
`field` (`ProjectV2FieldConfiguration!`)	The field that contains this value.
`milestone` (`Milestone`)	Milestone value of a field.

ProjectV2ItemFieldNumberValue

The value of a number field in a Project item.

`ProjectV2ItemFieldNumberValue` Implements

- `Node`
- `ProjectV2ItemFieldValueCommon`

Fields for `ProjectV2ItemFieldNumberValue`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`creator` (`Actor`)	The actor who created the item.
`databaseId` (`Int`)	Identifies the primary key from the database.
`field` (`ProjectV2FieldConfiguration!`)	The project field that contains this value.
`item` (`ProjectV2Item!`)	The project item that contains this value.
`number` (`Float`)	Number as a float(8).
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

ProjectV2ItemFieldPullRequestValue

The value of a pull request field in a Project item.

Fields for `ProjectV2ItemFieldPullRequestValue`

Name	Description
`field` (`ProjectV2FieldConfiguration!`)	The field that contains this value.
`pullRequests` (`PullRequestConnection`)	The pull requests for this field. **Arguments for `pullRequests`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`PullRequestOrder`) Ordering options for pull requests.

ProjectV2ItemFieldRepositoryValue

The value of a repository field in a Project item.

Fields for `ProjectV2ItemFieldRepositoryValue`

Name	Description
`field` (`ProjectV2FieldConfiguration!`)	The field that contains this value.
`repository` (`Repository`)	The repository for this field.

ProjectV2ItemFieldReviewerValue

The value of a reviewers field in a Project item.

Fields for `ProjectV2ItemFieldReviewerValue`

Name	Description
`field` (`ProjectV2FieldConfiguration!`)	The field that contains this value.
`reviewers` (`RequestedReviewerConnection`)	The reviewers for this field. **Arguments for `reviewers`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.

ProjectV2ItemFieldSingleSelectValue

The value of a single select field in a Project item.

`ProjectV2ItemFieldSingleSelectValue` Implements

- `Node`
- `ProjectV2ItemFieldValueCommon`

Fields for `ProjectV2ItemFieldSingleSelectValue`

Name	Description

`color` (`ProjectV2SingleSelectFieldOptionColor!`)	The color applied to the selected single-select option.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`creator` (`Actor`)	The actor who created the item.
`databaseId` (`Int`)	Identifies the primary key from the database.
`description` (`String`)	A plain-text description of the selected single-select option, such as what the option means.
`descriptionHTML` (`String`)	The description of the selected single-select option, including HTML tags.
`field` (`ProjectV2FieldConfiguration!`)	The project field that contains this value.
`item` (`ProjectV2Item!`)	The project item that contains this value.
`name` (`String`)	The name of the selected single select option.
`nameHTML` (`String`)	The html name of the selected single select option.
`optionId` (`String`)	The id of the selected single select option.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

ProjectV2ItemFieldTextValue

The value of a text field in a Project item.

`ProjectV2ItemFieldTextValue` Implements

- `Node`
- `ProjectV2ItemFieldValueCommon`

Fields for `ProjectV2ItemFieldTextValue`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`creator` (`Actor`)	The actor who created the item.
`databaseId` (`Int`)	Identifies the primary key from the database.
`field` (`ProjectV2FieldConfiguration!`)	The project field that contains this value.
`item` (`ProjectV2Item!`)	The project item that contains this value.
`text` (`String`)	Text value of a field.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

ProjectV2ItemFieldUserValue

The value of a user field in a Project item.

Fields for `ProjectV2ItemFieldUserValue`

Name	Description
`field` (`ProjectV2FieldConfiguration!`)	The field that contains this value.
`users` (`UserConnection`)	The users for this field. **Arguments for `users`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.

ProjectV2ItemFieldValueConnection

The connection type for ProjectV2ItemFieldValue.

Fields for `ProjectV2ItemFieldValueConnection`

Name	Description
`edges` (`[ProjectV2ItemFieldValueEdge]`)	A list of edges.
`nodes` (`[ProjectV2ItemFieldValue]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ProjectV2ItemFieldValueEdge

An edge in a connection.

Fields for `ProjectV2ItemFieldValueEdge`

Name	Description

`cursor` (`String!`)	A cursor for use in pagination.
`node` (`ProjectV2ItemFieldValue`)	The item at the end of the edge.

ProjectV2IterationField

An iteration field inside a project.

`ProjectV2IterationField` Implements

- `Node`
- `ProjectV2FieldCommon`

Fields for `ProjectV2IterationField`

Name	Description
`configuration` (`ProjectV2IterationFieldConfiguration!`)	Iteration configuration settings.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`dataType` (`ProjectV2FieldType!`)	The field's type.
`databaseId` (`Int`)	Identifies the primary key from the database.
`name` (`String!`)	The project field's name.
`project` (`ProjectV2!`)	The project that contains this field.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

ProjectV2IterationFieldConfiguration

Iteration field configuration for a project.

Fields for `ProjectV2IterationFieldConfiguration`

Name	Description
`completedIterations` (`[ProjectV2IterationFieldIteration!]!`)	The iteration's completed iterations.
`duration` (`Int!`)	The iteration's duration in days.
`iterations` (`[ProjectV2IterationFieldIteration!]!`)	The iteration's iterations.
`startDay` (`Int!`)	The iteration's start day of the week.

ProjectV2IterationFieldIteration

Iteration field iteration settings for a project.

Fields for `ProjectV2IterationFieldIteration`

Name	Description
`duration` (`Int!`)	The iteration's duration in days.
`id` (`String!`)	The iteration's ID.
`startDate` (`Date!`)	The iteration's start date.
`title` (`String!`)	The iteration's title.
`titleHTML` (`String!`)	The iteration's html title.

ProjectV2SingleSelectField

A single select field inside a project.

`ProjectV2SingleSelectField` Implements

- `Node`
- `ProjectV2FieldCommon`

Fields for `ProjectV2SingleSelectField`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`dataType` (`ProjectV2FieldType!`)	The field's type.
`databaseId` (`Int`)	Identifies the primary key from the database.
`name` (`String!`)	The project field's name.
`options` (`[ProjectV2SingleSelectFieldOption!]!`)	Options for the single select field. **Arguments for `options`** `names` (`[String!]`) Filter returned options to only those matching these names, case insensitive.
`project` (`ProjectV2!`)	The project that contains this field.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

ProjectV2SingleSelectFieldOption

Single select field option for a configuration for a project.

Fields for `ProjectV2SingleSelectFieldOption`

Name	Description
`color` (`ProjectV2SingleSelectFieldOptionColor!`)	The option's display color.
`description` (`String!`)	The option's plain-text description.
`descriptionHTML` (`String!`)	The option's description, possibly containing HTML.
`id` (`String!`)	The option's ID.
`name` (`String!`)	The option's name.
`nameHTML` (`String!`)	The option's html name.

ProjectV2SortBy

Represents a sort by field and direction.

Fields for `ProjectV2SortBy`

Name	Description
`direction` (`OrderDirection!`)	The direction of the sorting. Possible values are ASC and DESC.
`field` (`ProjectV2Field!`)	The field by which items are sorted.

ProjectV2SortByConnection

The connection type for ProjectV2SortBy.

Fields for `ProjectV2SortByConnection`

Name	Description
`edges` (`[ProjectV2SortByEdge]`)	A list of edges.
`nodes` (`[ProjectV2SortBy]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ProjectV2SortByEdge

An edge in a connection.

Fields for `ProjectV2SortByEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`ProjectV2SortBy`)	The item at the end of the edge.

ProjectV2SortByField

Represents a sort by field and direction.

Fields for `ProjectV2SortByField`

Name	Description
`direction` (`OrderDirection!`)	The direction of the sorting. Possible values are ASC and DESC.
`field` (`ProjectV2FieldConfiguration!`)	The field by which items are sorted.

ProjectV2SortByFieldConnection

The connection type for ProjectV2SortByField.

Fields for `ProjectV2SortByFieldConnection`

Name	Description
`edges` (`[ProjectV2SortByFieldEdge]`)	A list of edges.
`nodes` (`[ProjectV2SortByField]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ProjectV2SortByFieldEdge

An edge in a connection.

Fields for `ProjectV2SortByFieldEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`ProjectV2SortByField`)	The item at the end of the edge.

ProjectV2View

A view within a ProjectV2.

`ProjectV2View` Implements

- `Node`

Fields for `ProjectV2View`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`fields` (`ProjectV2FieldConfigurationConnection`)	The view's visible fields. **Arguments for `fields`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`ProjectV2FieldOrder`) Ordering options for the project v2 fields returned from the connection.
`filter` (`String`)	The project view's filter.
`groupBy` (`ProjectV2FieldConnection`)	The view's group-by field. **Deprecation notice** `groupBy` is deprecated. The `ProjectV2View#order_by` API is deprecated in favour of the more capable `ProjectV2View#group_by_field` API. Check out the `ProjectV2View#group_by_fields` API as an example for the more capable alternative. Removal on 2023-04-01 UTC. **Arguments for `groupBy`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ProjectV2FieldOrder`)
Ordering options for the project v2 fields returned from the connection.

`groupByFields` (`ProjectV2FieldConfigurationConnection`)	The view's group-by field.

Arguments for `groupByFields`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ProjectV2FieldOrder`)
Ordering options for the project v2 fields returned from the connection.

`layout` (`ProjectV2ViewLayout!`)	The project view's layout.
`name` (`String!`)	The project view's name.
`number` (`Int!`)	The project view's number.
`project` (`ProjectV2!`)	The project that contains this view.
`sortBy` (`ProjectV2SortByConnection`)	The view's sort-by config.

Deprecation notice
`sortBy` is deprecated.
The `ProjectV2View#sort_by` API is deprecated in favour of the more capable `ProjectV2View#sort_by_fields` API. Check out the `ProjectV2View#sort_by_fields` API

as an example for the more capable alternative. Removal on 2023-04-01 UTC.

Arguments for `sortBy`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`sortByFields` (`ProjectV2SortByFieldConnection`)

The view's sort-by config.

Arguments for `sortByFields`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`updatedAt` (`DateTime!`)

Identifies the date and time when the object was last updated.

`verticalGroupBy` (`ProjectV2FieldConnection`)

The view's vertical-group-by field.

Deprecation notice
`verticalGroupBy` is deprecated.
The `ProjectV2View#vertical_group_by` API is deprecated in favour of the more capable `ProjectV2View#vertical_group_by_fields` API. Check out the `ProjectV2View#vertical_group_by_fields` API as an example for the more capable alternative. Removal on 2023-04-01 UTC.

Arguments for `verticalGroupBy`

`after` (`String`)
Returns the elements in the list that come

after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ProjectV2FieldOrder`)
Ordering options for the project v2 fields returned from the connection.

`verticalGroupByFields` (`ProjectV2FieldConfigurationConnection`)

The view's vertical-group-by field.

Arguments for `verticalGroupByFields`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ProjectV2FieldOrder`)
Ordering options for the project v2 fields returned from the connection.

`visibleFields` (`ProjectV2FieldConnection`)

The view's visible fields.

Deprecation notice
`visibleFields` is deprecated.
The `ProjectV2View#visibleFields` API is deprecated in favour of the more capable `ProjectV2View#fields` API. Check out the `ProjectV2View#fields` API as an example for the more capable alternative. Removal on 2023-01-01 UTC.

Arguments for `visibleFields`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ProjectV2FieldOrder`)
Ordering options for the project v2 fields returned from the connection.

ProjectV2ViewConnection

The connection type for ProjectV2View.

Fields for `ProjectV2ViewConnection`

Name	Description
`edges` (`[ProjectV2ViewEdge]`)	A list of edges.
`nodes` (`[ProjectV2View]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ProjectV2ViewEdge

An edge in a connection.

Fields for `ProjectV2ViewEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`ProjectV2View`)	The item at the end of the edge.

ProjectV2Workflow

A workflow inside a project.

`ProjectV2Workflow` Implements

- `Node`

Fields for `ProjectV2Workflow`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`enabled` (`Boolean!`)	The workflows' enabled state.
`name` (`String!`)	The workflows' name.

`number` (`Int!`)	The workflows' number.
`project` (`ProjectV2!`)	The project that contains this workflow.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

ProjectV2WorkflowConnection

The connection type for ProjectV2Workflow.

Fields for `ProjectV2WorkflowConnection`

Name	Description
`edges` (`[ProjectV2WorkflowEdge]`)	A list of edges.
`nodes` (`[ProjectV2Workflow]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ProjectV2WorkflowEdge

An edge in a connection.

Fields for `ProjectV2WorkflowEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`ProjectV2Workflow`)	The item at the end of the edge.

PublicKey

A user's public key.

`PublicKey` Implements

- `Node`

Fields for `PublicKey`

Name	Description
`accessedAt` (`DateTime`)	The last time this authorization was used to perform an action. Values will be null for keys not owned by the user.
`createdAt` (`DateTime`)	Identifies the date and time when the key was created. Keys created before March 5th, 2014

	created. Keys created before March 5th, 2014 have inaccurate values. Values will be null for keys not owned by the user.
`fingerprint` (`String!`)	The fingerprint for this PublicKey.
`isReadOnly` (`Boolean`)	Whether this PublicKey is read-only or not. Values will be null for keys not owned by the user.
`key` (`String!`)	The public key string.
`updatedAt` (`DateTime`)	Identifies the date and time when the key was updated. Keys created before March 5th, 2014 may have inaccurate values. Values will be null for keys not owned by the user.

PublicKeyConnection

The connection type for PublicKey.

Fields for `PublicKeyConnection`

Name	Description
`edges` (`[PublicKeyEdge]`)	A list of edges.
`nodes` (`[PublicKey]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

PublicKeyEdge

An edge in a connection.

Fields for `PublicKeyEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`PublicKey`)	The item at the end of the edge.

PullRequest

A repository pull request.

`PullRequest` Implements

- `Assignable`
- `Closable`

- `Comment`
- `Labelable`
- `Lockable`
- `Node`
- `ProjectV2Owner`
- `Reactable`
- `RepositoryNode`
- `Subscribable`
- `UniformResourceLocatable`
- `Updatable`
- `UpdatableComment`

Fields for `PullRequest`

Name	Description
`activeLockReason` (`LockReason`)	Reason that the conversation was locked.
`additions` (`Int!`)	The number of additions in this pull request.
`assignees` (`UserConnection!`)	A list of Users assigned to this object. **Arguments for `assignees`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`author` (`Actor`)	The actor who authored the comment.
`authorAssociation` (`CommentAuthorAssociation!`)	Author's association with the subject of the comment.
`autoMergeRequest` (`AutoMergeRequest`)	Returns the auto-merge request object if one exists for this pull request.
`baseRef` (`Ref`)	Identifies the base Ref associated with the pull request.
`baseRefName` (`String!`)	Identifies the name of the base Ref associated with the pull request, even if the ref has been deleted.
`baseRefOid` (`GitObjectID!`)	Identifies the oid of the base ref associated with the pull request, even if the ref has been deleted.

`baseRepository` (`Repository`)	The repository associated with this pull request's base Ref.
`body` (`String!`)	The body as Markdown.
`bodyHTML` (`HTML!`)	The body rendered to HTML.
`bodyText` (`String!`)	The body rendered to text.
`canBeRebased` (`Boolean!`)	Whether or not the pull request is rebaseable. **Preview notice** `canBeRebased` is available under the Merge info preview more detailed information about a pull request's merge state preview. During the preview period, the API may change without notice.
`changedFiles` (`Int!`)	The number of changed files in this pull request.
`checksResourcePath` (`URI!`)	The HTTP path for the checks of this pull request.
`checksUrl` (`URI!`)	The HTTP URL for the checks of this pull request.
`closed` (`Boolean!`)	`true` if the pull request is closed.
`closedAt` (`DateTime`)	Identifies the date and time when the object was closed.
`closingIssuesReferences` (`IssueConnection`)	List of issues that were may be closed by this pull request. **Arguments for `closingIssuesReferences`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`IssueOrder`) Ordering options for issues returned from the connection. `userLinkedOnly` (`Boolean`) Return only manually linked Issues. The default value is `false`.
`comments` (`IssueCommentConnection!`)	A list of comments associated with the pull request.

Arguments for `comments`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`IssueCommentOrder`)
Ordering options for issue comments returned from the connection.

`commits` (`PullRequestCommitConnection!`)

A list of commits present in this pull request's head branch not present in the base branch.

Arguments for `commits`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`createdAt` (`DateTime!`)

Identifies the date and time when the object was created.

`createdViaEmail` (`Boolean!`)

Check if this comment was created via an email reply.

`databaseId` (`Int`)

Identifies the primary key from the database.

`deletions` (`Int!`)

The number of deletions in this pull request.

`editor` (`Actor`)

The actor who edited this pull request's body.

`files` (`PullRequestChangedFileConnection`)

Lists the files changed within this pull request.

Arguments for `files`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`headRef` (`Ref`)	Identifies the head Ref associated with the pull request.
`headRefName` (`String!`)	Identifies the name of the head Ref associated with the pull request, even if the ref has been deleted.
`headRefOid` (`GitObjectID!`)	Identifies the oid of the head ref associated with the pull request, even if the ref has been deleted.
`headRepository` (`Repository`)	The repository associated with this pull request's head Ref.
`headRepositoryOwner` (`RepositoryOwner`)	The owner of the repository associated with this pull request's head Ref.
`hovercard` (`Hovercard!`)	The hovercard information for this issue.

Arguments for `hovercard`

`includeNotificationContexts` (`Boolean`)
Whether or not to include notification contexts.
The default value is `true` .

`includesCreatedEdit` (`Boolean!`)	Check if this comment was edited and includes an edit with the creation data.
`isCrossRepository` (`Boolean!`)	The head and base repositories are different.
`isDraft` (`Boolean!`)	Identifies if the pull request is a draft.
`isReadByViewer` (`Boolean`)	Is this pull request read by the viewer.
`labels` (`LabelConnection`)	A list of labels associated with the object.

Arguments for `labels`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)

Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`LabelOrder`)
Ordering options for labels returned from the connection.

`lastEditedAt` (`DateTime`)	The moment the editor made the last edit.
`latestOpinionatedReviews` (`PullRequestReviewConnection`)	A list of latest reviews per user associated with the pull request. **Arguments for `latestOpinionatedReviews`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `writersOnly` (`Boolean`) Only return reviews from user who have write access to the repository. The default value is `false`.
`latestReviews` (`PullRequestReviewConnection`)	A list of latest reviews per user associated with the pull request that are not also pending review. **Arguments for `latestReviews`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`locked` (`Boolean!`)	`true` if the pull request is locked.

`maintainerCanModify` (`Boolean!`)	Indicates whether maintainers can modify the pull request.
`mergeCommit` (`Commit`)	The commit that was created when this pull request was merged.
`mergeQueueEntry` (`MergeQueueEntry`)	The merge queue entry of the pull request in the base branch's merge queue.
`mergeStateStatus` (`MergeStateStatus!`)	Detailed information about the current pull request merge state status. **Preview notice** `mergeStateStatus` is available under the Merge info preview more detailed information about a pull request's merge state preview. During the preview period, the API may change without notice.
`mergeable` (`MergeableState!`)	Whether or not the pull request can be merged based on the existence of merge conflicts.
`merged` (`Boolean!`)	Whether or not the pull request was merged.
`mergedAt` (`DateTime`)	The date and time that the pull request was merged.
`mergedBy` (`Actor`)	The actor who merged the pull request.
`milestone` (`Milestone`)	Identifies the milestone associated with the pull request.
`number` (`Int!`)	Identifies the pull request number.
`participants` (`UserConnection!`)	A list of Users that are participating in the Pull Request conversation. **Arguments for `participants`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`permalink` (`URI!`)	The permalink to the pull request.
`potentialMergeCommit` (`Commit`)	The commit that GitHub automatically generated to test if this pull request could be merged. This field will not return a value if the pull request is merged, or if the test merge commit is still being generated. See the

`mergeable` field for more details on the mergeability of the pull request.

`projectCards` (`ProjectCardConnection!`)

List of project cards associated with this pull request.

Arguments for `projectCards`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`archivedStates` (`[ProjectCardArchivedState]`)
A list of archived states to filter the cards by.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`projectItems` (`ProjectV2ItemConnection!`)

List of project items associated with this pull request.

Arguments for `projectItems`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`includeArchived` (`Boolean`)
Include archived items.
The default value is `true` .

`last` (`Int`)
Returns the last *n* elements from the list.

`projectV2` (`ProjectV2`)

Find a project by number.

Arguments for `projectV2`

`number` (`Int!`)
The project number.

`projectsV2` (`ProjectV2Connection!`)	A list of projects under the owner. **Arguments for `projectsV2`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`ProjectV2Order`) How to order the returned projects. `query` (`String`) A project to search for under the the owner.
`publishedAt` (`DateTime`)	Identifies when the comment was published at.
`reactionGroups` (`[ReactionGroup!]`)	A list of reactions grouped by content left on the subject.
`reactions` (`ReactionConnection!`)	A list of Reactions left on the Issue. **Arguments for `reactions`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `content` (`ReactionContent`) Allows filtering Reactions by emoji. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`ReactionOrder`) Allows specifying the order in which reactions are returned.
`repository` (`Repository!`)	The repository associated with this node.
`resourcePath` (`URI!`)	The HTTP path for this pull request.
`revertResourcePath` (`URI!`)	The HTTP path for reverting this pull request.

`revertUrl` (`URI!`)	The HTTP URL for reverting this pull request.
`reviewDecision` (`PullRequestReviewDecision`)	The current status of this pull request with respect to code review.
`reviewRequests` (`ReviewRequestConnection`)	A list of review requests associated with the pull request. **Arguments for `reviewRequests`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`reviewThreads` (`PullRequestReviewThreadConnection!`)	The list of all review threads for this pull request. **Arguments for `reviewThreads`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`reviews` (`PullRequestReviewConnection`)	A list of reviews associated with the pull request. **Arguments for `reviews`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `author` (`String`) Filter by author of the review. `before` (`String`) Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`states` (`[PullRequestReviewState!]`)
A list of states to filter the reviews.

`state` (`PullRequestState!`)	Identifies the state of the pull request.
`suggestedReviewers` (`[SuggestedReviewer]!`)	A list of reviewer suggestions based on commit history and past review comments.
`timeline` (`PullRequestTimelineConnection!`)	A list of events, comments, commits, etc. associated with the pull request.

Deprecation notice
`timeline` is deprecated.
`timeline` will be removed Use PullRequest.timelineItems instead. Removal on 2020-10-01 UTC.

Arguments for `timeline`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`since` (`DateTime`)
Allows filtering timeline events by a `since` timestamp.

`timelineItems` (`PullRequestTimelineItemsConnection!`)	A list of events, comments, commits, etc. associated with the pull request.

Arguments for `timelineItems`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)

Returns the first *n* elements from the list.

`itemTypes` (`[PullRequestTimelineItemsItemType!]`)
Filter timeline items by type.

`last` (`Int`)
Returns the last *n* elements from the list.

`since` (`DateTime`)
Filter timeline items by a `since` timestamp.

`skip` (`Int`)
Skips the first *n* elements in the list.

`title` (`String!`)	Identifies the pull request title.
`titleHTML` (`HTML!`)	Identifies the pull request title rendered to HTML.
`totalCommentsCount` (`Int`)	Returns a count of how many comments this pull request has received.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this pull request.
`userContentEdits` (`UserContentEditConnection`)	A list of edits to this content.

Arguments for `userContentEdits`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`viewerCanApplySuggestion` (`Boolean!`)	Whether or not the viewer can apply suggestion.
`viewerCanClose` (`Boolean!`)	Indicates if the object can be closed by the viewer.
`viewerCanDeleteHeadRef` (`Boolean!`)	Check if the viewer can restore the deleted head ref.
`viewerCanDisableAutoMerge` (`Boolean!`)	Whether or not the viewer can disable auto-merge.
`viewerCanEditFiles` (`Boolean!`)	Can the viewer edit files within this pull request.

`viewerCanEnableAutoMerge` (`Boolean!`)	Whether or not the viewer can enable auto-merge.
`viewerCanMergeAsAdmin` (`Boolean!`)	Indicates whether the viewer can bypass branch protections and merge the pull request immediately.
`viewerCanReact` (`Boolean!`)	Can user react to this subject.
`viewerCanReopen` (`Boolean!`)	Indicates if the object can be reopened by the viewer.
`viewerCanSubscribe` (`Boolean!`)	Check if the viewer is able to change their subscription status for the repository.
`viewerCanUpdate` (`Boolean!`)	Check if the current viewer can update this object.
`viewerCanUpdateBranch` (`Boolean!`)	Whether or not the viewer can update the head ref of this PR, by merging or rebasing the base ref. If the head ref is up to date or unable to be updated by this user, this will return false.
`viewerCannotUpdateReasons` (`[CommentCannotUpdateReason!]!`)	Reasons why the current viewer can not update this comment.
`viewerDidAuthor` (`Boolean!`)	Did the viewer author this comment.
`viewerLatestReview` (`PullRequestReview`)	The latest review given from the viewer.
`viewerLatestReviewRequest` (`ReviewRequest`)	The person who has requested the viewer for review on this pull request.
`viewerMergeBodyText` (`String!`)	The merge body text for the viewer and method. **Arguments for `viewerMergeBodyText`** `mergeType` (`PullRequestMergeMethod`) The merge method for the message.
`viewerMergeHeadlineText` (`String!`)	The merge headline text for the viewer and method. **Arguments for `viewerMergeHeadlineText`** `mergeType` (`PullRequestMergeMethod`) The merge method for the message.
`viewerSubscription` (`SubscriptionState`)	Identifies if the viewer is watching, not watching, or ignoring the subscribable entity.

PullRequestChangedFile

A file changed in a pull request.

Fields for `PullRequestChangedFile`

Name	Description
`additions` (`Int!`)	The number of additions to the file.
`changeType` (`PatchStatus!`)	How the file was changed in this PullRequest.
`deletions` (`Int!`)	The number of deletions to the file.
`path` (`String!`)	The path of the file.
`viewerViewedState` (`FileViewedState!`)	The state of the file for the viewer.

PullRequestChangedFileConnection

The connection type for PullRequestChangedFile.

Fields for `PullRequestChangedFileConnection`

Name	Description
`edges` (`[PullRequestChangedFileEdge]`)	A list of edges.
`nodes` (`[PullRequestChangedFile]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

PullRequestChangedFileEdge

An edge in a connection.

Fields for `PullRequestChangedFileEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`PullRequestChangedFile`)	The item at the end of the edge.

PullRequestCommit

Represents a Git commit part of a pull request.

`PullRequestCommit` Implements

- `Node`
- `UniformResourceLocatable`

Fields for `PullRequestCommit`

Name	Description
`commit` (`Commit!`)	The Git commit object.
`pullRequest` (`PullRequest!`)	The pull request this commit belongs to.
`resourcePath` (`URI!`)	The HTTP path for this pull request commit.
`url` (`URI!`)	The HTTP URL for this pull request commit.

PullRequestCommitCommentThread

Represents a commit comment thread part of a pull request.

`PullRequestCommitCommentThread` Implements

- `Node`
- `RepositoryNode`

Fields for `PullRequestCommitCommentThread`

Name	Description
`comments` (`CommitCommentConnection!`)	The comments that exist in this thread. **Arguments for `comments`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`commit` (`Commit!`)	The commit the comments were made on.
`path` (`String`)	The file the comments were made on.
`position` (`Int`)	The position in the diff for the commit that the comment was made on.
`pullRequest` (`PullRequest!`)	The pull request this commit comment thread belongs to.
`repository` (`Repository!`)	The repository associated with this node.

PullRequestCommitConnection

The connection type for PullRequestCommit.

Fields for `PullRequestCommitConnection`

Name	Description
`edges` (`[PullRequestCommitEdge]`)	A list of edges.
`nodes` (`[PullRequestCommit]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

PullRequestCommitEdge

An edge in a connection.

Fields for `PullRequestCommitEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`PullRequestCommit`)	The item at the end of the edge.

PullRequestConnection

The connection type for PullRequest.

Fields for `PullRequestConnection`

Name	Description
`edges` (`[PullRequestEdge]`)	A list of edges.
`nodes` (`[PullRequest]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

PullRequestContributionsByRepository

This aggregates pull requests opened by a user within one repository.

Fields for `PullRequestContributionsByRepository`

Name	Description
`contributions` (`CreatedPullRequestContributionConnection!`)	The pull request contributions. **Arguments for `contributions`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`ContributionOrder`) Ordering options for contributions returned from the connection.
`repository` (`Repository!`)	The repository in which the pull requests were opened.

PullRequestEdge

An edge in a connection.

Fields for `PullRequestEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`PullRequest`)	The item at the end of the edge.

PullRequestParameters

Require all commits be made to a non-target branch and submitted via a pull request before they can be merged.

Fields for `PullRequestParameters`

Name	Description
`dismissStaleReviewsOnPush` (`Boolean!`)	New, reviewable commits pushed will dismiss previous pull request review approvals.
`requireCodeOwnerReview` (`Boolean!`)	Require an approving review in pull requests that modify files that have a designated code owner.
`requireLastPushApproval` (`Boolean!`)	Whether the most recent reviewable push must

[illegible] ([illegible])	[illegible] be approved by someone other than the person who pushed it.
`requiredApprovingReviewCount` (`Int!`)	The number of approving reviews that are required before a pull request can be merged.
`requiredReviewThreadResolution` (`Boolean!`)	All conversations on code must be resolved before a pull request can be merged.

PullRequestReview

A review object for a given pull request.

`PullRequestReview` Implements

- `Comment`
- `Deletable`
- `Node`
- `Reactable`
- `RepositoryNode`
- `Updatable`
- `UpdatableComment`

Fields for `PullRequestReview`

Name	Description
`author` (`Actor`)	The actor who authored the comment.
`authorAssociation` (`CommentAuthorAssociation!`)	Author's association with the subject of the comment.
`authorCanPushToRepository` (`Boolean!`)	Indicates whether the author of this review has push access to the repository.
`body` (`String!`)	Identifies the pull request review body.
`bodyHTML` (`HTML!`)	The body rendered to HTML.
`bodyText` (`String!`)	The body of this review rendered as plain text.
`comments` (`PullRequestReviewCommentConnection!`)	A list of review comments for the current pull request review. **Arguments for `comments`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`)

`last` (`Int`)
Returns the last *n* elements from the list.

`commit` (`Commit`)	Identifies the commit associated with this pull request review.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`createdViaEmail` (`Boolean!`)	Check if this comment was created via an email reply.
`databaseId` (`Int`)	Identifies the primary key from the database.
`editor` (`Actor`)	The actor who edited the comment.
`includesCreatedEdit` (`Boolean!`)	Check if this comment was edited and includes an edit with the creation data.
`lastEditedAt` (`DateTime`)	The moment the editor made the last edit.
`onBehalfOf` (`TeamConnection!`)	A list of teams that this review was made on behalf of.

Arguments for `onBehalfOf`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`publishedAt` (`DateTime`)	Identifies when the comment was published at.
`pullRequest` (`PullRequest!`)	Identifies the pull request associated with this pull request review.
`reactionGroups` (`[ReactionGroup!]`)	A list of reactions grouped by content left on the subject.
`reactions` (`ReactionConnection!`)	A list of Reactions left on the Issue.

Arguments for `reactions`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`content` (`ReactionContent`)
Allows filtering Reactions by emoji.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ReactionOrder`)
Allows specifying the order in which reactions are returned.

`repository` (`Repository!`)	The repository associated with this node.
`resourcePath` (`URI!`)	The HTTP path permalink for this PullRequestReview.
`state` (`PullRequestReviewState!`)	Identifies the current state of the pull request review.
`submittedAt` (`DateTime`)	Identifies when the Pull Request Review was submitted.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL permalink for this PullRequestReview.
`userContentEdits` (`UserContentEditConnection`)	A list of edits to this content.

Arguments for `userContentEdits`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`viewerCanDelete` (`Boolean!`)	Check if the current viewer can delete this object.
`viewerCanReact` (`Boolean!`)	Can user react to this subject.
`viewerCanUpdate` (`Boolean!`)	Check if the current viewer can update this object.
`viewerCannotUpdateReasons` (`[CommentCannotUpdateReason!]!`)	Reasons why the current viewer can not update this comment.

`viewerDidAuthor` (`Boolean!`)	Did the viewer author this comment.

PullRequestReviewComment

A review comment associated with a given repository pull request.

`PullRequestReviewComment` Implements

- `Comment`
- `Deletable`
- `Minimizable`
- `Node`
- `Reactable`
- `RepositoryNode`
- `Updatable`
- `UpdatableComment`

Fields for `PullRequestReviewComment`

Name	Description
`author` (`Actor`)	The actor who authored the comment.
`authorAssociation` (`CommentAuthorAssociation!`)	Author's association with the subject of the comment.
`body` (`String!`)	The comment body of this review comment.
`bodyHTML` (`HTML!`)	The body rendered to HTML.
`bodyText` (`String!`)	The comment body of this review comment rendered as plain text.
`commit` (`Commit`)	Identifies the commit associated with the comment.
`createdAt` (`DateTime!`)	Identifies when the comment was created.
`createdViaEmail` (`Boolean!`)	Check if this comment was created via an email reply.
`databaseId` (`Int`)	Identifies the primary key from the database.
`diffHunk` (`String!`)	The diff hunk to which the comment applies.
`draftedAt` (`DateTime!`)	Identifies when the comment was created in a draft state.
`editor` (`Actor`)	The actor who edited the comment.
`includesCreatedEdit` (`Boolean!`)	Check if this comment was edited and includes an edit with the creation data.
`isMinimized` (`Boolean!`)	Returns whether or not a comment has been minimized.
`lastEditedAt` (`DateTime`)	The moment the editor made the last edit.

`line` (`Int`)	The end line number on the file to which the comment applies.
`minimizedReason` (`String`)	Returns why the comment was minimized. One of `abuse`, `off-topic`, `outdated`, `resolved`, `duplicate` and `spam`. Note that the case and formatting of these values differs from the inputs to the `MinimizeComment` mutation.
`originalCommit` (`Commit`)	Identifies the original commit associated with the comment.
`originalLine` (`Int`)	The end line number on the file to which the comment applied when it was first created.
`originalPosition` (`Int!`)	The original line index in the diff to which the comment applies. **Deprecation notice** `originalPosition` is deprecated. We are phasing out diff-relative positioning for PR comments Removal on 2023-10-01 UTC.
`originalStartLine` (`Int`)	The start line number on the file to which the comment applied when it was first created.
`outdated` (`Boolean!`)	Identifies when the comment body is outdated.
`path` (`String!`)	The path to which the comment applies.
`position` (`Int`)	The line index in the diff to which the comment applies. **Deprecation notice** `position` is deprecated. We are phasing out diff-relative positioning for PR comments Use the `line` and `startLine` fields instead, which are file line numbers instead of diff line numbers Removal on 2023-10-01 UTC.
`publishedAt` (`DateTime`)	Identifies when the comment was published at.
`pullRequest` (`PullRequest!`)	The pull request associated with this review comment.
`pullRequestReview` (`PullRequestReview`)	The pull request review associated with this review comment.
`reactionGroups` (`[ReactionGroup!]`)	A list of reactions grouped by content left on the subject.
`reactions` (`ReactionConnection!`)	A list of Reactions left on the Issue. **Arguments for `reactions`**

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`content` (`ReactionContent`)
Allows filtering Reactions by emoji.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ReactionOrder`)
Allows specifying the order in which reactions are returned.

`replyTo` (`PullRequestReviewComment`)	The comment this is a reply to.
`repository` (`Repository!`)	The repository associated with this node.
`resourcePath` (`URI!`)	The HTTP path permalink for this review comment.
`startLine` (`Int`)	The start line number on the file to which the comment applies.
`state` (`PullRequestReviewCommentState!`)	Identifies the state of the comment.
`subjectType` (`PullRequestReviewThreadSubjectType!`)	The level at which the comments in the corresponding thread are targeted, can be a diff line or a file.
`updatedAt` (`DateTime!`)	Identifies when the comment was last updated.
`url` (`URI!`)	The HTTP URL permalink for this review comment.
`userContentEdits` (`UserContentEditConnection`)	A list of edits to this content.

Arguments for `userContentEdits`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`viewerCanDelete` (`Boolean!`)	Check if the current viewer can delete this object.
`viewerCanMinimize` (`Boolean!`)	Check if the current viewer can minimize this object.
`viewerCanReact` (`Boolean!`)	Can user react to this subject.
`viewerCanUpdate` (`Boolean!`)	Check if the current viewer can update this object.
`viewerCannotUpdateReasons` (`[CommentCannotUpdateReason!]!`)	Reasons why the current viewer can not update this comment.
`viewerDidAuthor` (`Boolean!`)	Did the viewer author this comment.

PullRequestReviewCommentConnection

The connection type for PullRequestReviewComment.

Fields for `PullRequestReviewCommentConnection`

Name	Description
`edges` (`[PullRequestReviewCommentEdge]`)	A list of edges.
`nodes` (`[PullRequestReviewComment]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

PullRequestReviewCommentEdge

An edge in a connection.

Fields for `PullRequestReviewCommentEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`PullRequestReviewComment`)	The item at the end of the edge.

PullRequestReviewConnection

The connection type for PullRequestReview.

Fields for `PullRequestReviewConnection`

Name	Description
`edges` (`[PullRequestReviewEdge]`)	A list of edges.

`nodes` (`[PullRequestReview]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

PullRequestReviewContributionsByRepository

This aggregates pull request reviews made by a user within one repository.

Fields for `PullRequestReviewContributionsByRepository`

Name	Description
`contributions` (`CreatedPullRequestReviewContributionConnection!`)	The pull request review contributions. **Arguments for `contributions`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`ContributionOrder`) Ordering options for contributions returned from the connection.
`repository` (`Repository!`)	The repository in which the pull request reviews were made.

PullRequestReviewEdge

An edge in a connection.

Fields for `PullRequestReviewEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`PullRequestReview`)	The item at the end of the edge.

PullRequestReviewThread

A threaded list of comments for a given pull request.

`PullRequestReviewThread` Implements

- `Node`

Fields for `PullRequestReviewThread`

Name	Description
`comments` (`PullRequestReviewCommentConnection!`)	A list of pull request comments associated with the thread. **Arguments for `comments`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `skip` (`Int`) Skips the first *n* elements in the list.
`diffSide` (`DiffSide!`)	The side of the diff on which this thread was placed.
`isCollapsed` (`Boolean!`)	Whether or not the thread has been collapsed (resolved).
`isOutdated` (`Boolean!`)	Indicates whether this thread was outdated by newer changes.
`isResolved` (`Boolean!`)	Whether this thread has been resolved.
`line` (`Int`)	The line in the file to which this thread refers.
`originalLine` (`Int`)	The original line in the file to which this thread refers.
`originalStartLine` (`Int`)	The original start line in the file to which this thread refers (multi-line only).
`path` (`String!`)	Identifies the file path of this thread.
`pullRequest` (`PullRequest!`)	Identifies the pull request associated with this thread.

`repository` (`Repository!`)	Identifies the repository associated with this thread.
`resolvedBy` (`User`)	The user who resolved this thread.
`startDiffSide` (`DiffSide`)	The side of the diff that the first line of the thread starts on (multi-line only).
`startLine` (`Int`)	The start line in the file to which this thread refers (multi-line only).
`subjectType` (`PullRequestReviewThreadSubjectType!`)	The level at which the comments in the corresponding thread are targeted, can be a diff line or a file.
`viewerCanReply` (`Boolean!`)	Indicates whether the current viewer can reply to this thread.
`viewerCanResolve` (`Boolean!`)	Whether or not the viewer can resolve this thread.
`viewerCanUnresolve` (`Boolean!`)	Whether or not the viewer can unresolve this thread.

PullRequestReviewThreadConnection

Review comment threads for a pull request review.

Fields for `PullRequestReviewThreadConnection`

Name	Description
`edges` (`[PullRequestReviewThreadEdge]`)	A list of edges.
`nodes` (`[PullRequestReviewThread]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

PullRequestReviewThreadEdge

An edge in a connection.

Fields for `PullRequestReviewThreadEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`PullRequestReviewThread`)	The item at the end of the edge.

PullRequestRevisionMarker

Represents the latest point in the pull request timeline for which the viewer has seen the pull request's commits.

Fields for `PullRequestRevisionMarker`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`lastSeenCommit` (`Commit!`)	The last commit the viewer has seen.
`pullRequest` (`PullRequest!`)	The pull request to which the marker belongs.

PullRequestTemplate

A repository pull request template.

Fields for `PullRequestTemplate`

Name	Description
`body` (`String`)	The body of the template.
`filename` (`String`)	The filename of the template.
`repository` (`Repository!`)	The repository the template belongs to.

PullRequestThread

A threaded list of comments for a given pull request.

`PullRequestThread` Implements

- `Node`

Fields for `PullRequestThread`

Name	Description
`comments` (`PullRequestReviewCommentConnection!`)	A list of pull request comments associated with the thread. **Arguments for `comments`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`skip` (`Int`)
Skips the first *n* elements in the list.

`diffSide` (`DiffSide!`)	The side of the diff on which this thread was placed.
`isCollapsed` (`Boolean!`)	Whether or not the thread has been collapsed (resolved).
`isOutdated` (`Boolean!`)	Indicates whether this thread was outdated by newer changes.
`isResolved` (`Boolean!`)	Whether this thread has been resolved.
`line` (`Int`)	The line in the file to which this thread refers.
`path` (`String!`)	Identifies the file path of this thread.
`pullRequest` (`PullRequest!`)	Identifies the pull request associated with this thread.
`repository` (`Repository!`)	Identifies the repository associated with this thread.
`resolvedBy` (`User`)	The user who resolved this thread.
`startDiffSide` (`DiffSide`)	The side of the diff that the first line of the thread starts on (multi-line only).
`startLine` (`Int`)	The line of the first file diff in the thread.
`subjectType` (`PullRequestReviewThreadSubjectType!`)	The level at which the comments in the corresponding thread are targeted, can be a diff line or a file.
`viewerCanReply` (`Boolean!`)	Indicates whether the current viewer can reply to this thread.
`viewerCanResolve` (`Boolean!`)	Whether or not the viewer can resolve this thread.
`viewerCanUnresolve` (`Boolean!`)	Whether or not the viewer can unresolve this thread.

PullRequestTimelineConnection

The connection type for PullRequestTimelineItem.

Fields for `PullRequestTimelineConnection`

Name	Description
`edges` (`[PullRequestTimelineItemEdge]`)	A list of edges.
`nodes` (`[PullRequestTimelineItem]`)	A list of nodes.

`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

PullRequestTimelineItemEdge

An edge in a connection.

Fields for `PullRequestTimelineItemEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`PullRequestTimelineItem`)	The item at the end of the edge.

PullRequestTimelineItemsConnection

The connection type for PullRequestTimelineItems.

Fields for `PullRequestTimelineItemsConnection`

Name	Description
`edges` (`[PullRequestTimelineItemsEdge]`)	A list of edges.
`filteredCount` (`Int!`)	Identifies the count of items after applying `before` and `after` filters.
`nodes` (`[PullRequestTimelineItems]`)	A list of nodes.
`pageCount` (`Int!`)	Identifies the count of items after applying `before` / `after` filters and `first` / `last` / `skip` slicing.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.
`updatedAt` (`DateTime!`)	Identifies the date and time when the timeline was last updated.

PullRequestTimelineItemsEdge

An edge in a connection.

Fields for `PullRequestTimelineItemsEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.

`node` (`PullRequestTimelineItems`)	The item at the end of the edge.

Push

A Git push.

`Push` Implements

- `Node`

Fields for `Push`

Name	Description
`nextSha` (`GitObjectID`)	The SHA after the push.
`permalink` (`URI!`)	The permalink for this push.
`previousSha` (`GitObjectID`)	The SHA before the push.
`pusher` (`Actor!`)	The actor who pushed.
`repository` (`Repository!`)	The repository that was pushed to.

PushAllowance

A team, user, or app who has the ability to push to a protected branch.

`PushAllowance` Implements

- `Node`

Fields for `PushAllowance`

Name	Description
`actor` (`PushAllowanceActor`)	The actor that can push.
`branchProtectionRule` (`BranchProtectionRule`)	Identifies the branch protection rule associated with the allowed user, team, or app.

PushAllowanceConnection

The connection type for PushAllowance.

Fields for `PushAllowanceConnection`

Name	Description
`edges` (`[PushAllowanceEdge]`)	A list of edges.
`nodes` (`[PushAllowance]`)	A list of nodes.

`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

PushAllowanceEdge

An edge in a connection.

Fields for `PushAllowanceEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`PushAllowance`)	The item at the end of the edge.

RateLimit

Represents the client's rate limit.

Fields for `RateLimit`

Name	Description
`cost` (`Int!`)	The point cost for the current query counting against the rate limit.
`limit` (`Int!`)	The maximum number of points the client is permitted to consume in a 60 minute window.
`nodeCount` (`Int!`)	The maximum number of nodes this query may return.
`remaining` (`Int!`)	The number of points remaining in the current rate limit window.
`resetAt` (`DateTime!`)	The time at which the current rate limit window resets in UTC epoch seconds.
`used` (`Int!`)	The number of points used in the current rate limit window.

ReactingUserConnection

The connection type for User.

Fields for `ReactingUserConnection`

Name	Description
`edges` (`[ReactingUserEdge]`)	A list of edges.
`nodes` (`[User]`)	A list of nodes.

`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ReactingUserEdge

Represents a user that's made a reaction.

Fields for `ReactingUserEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`reactedAt` (`DateTime!`)	The moment when the user made the reaction.

Reaction

An emoji reaction to a particular piece of content.

`Reaction` Implements

- `Node`

Fields for `Reaction`

Name	Description
`content` (`ReactionContent!`)	Identifies the emoji reaction.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`reactable` (`Reactable!`)	The reactable piece of content.
`user` (`User`)	Identifies the user who created this reaction.

ReactionConnection

A list of reactions that have been left on the subject.

Fields for `ReactionConnection`

Name	Description
`edges` (`[ReactionEdge]`)	A list of edges.
`nodes` (`[Reaction]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.

`totalCount` (`Int!`)	Identifies the total count of items in the connection.
`viewerHasReacted` (`Boolean!`)	Whether or not the authenticated user has left a reaction on the subject.

ReactionEdge

An edge in a connection.

Fields for `ReactionEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Reaction`)	The item at the end of the edge.

ReactionGroup

A group of emoji reactions to a particular piece of content.

Fields for `ReactionGroup`

Name	Description
`content` (`ReactionContent!`)	Identifies the emoji reaction.
`createdAt` (`DateTime`)	Identifies when the reaction was created.
`reactors` (`ReactorConnection!`)	Reactors to the reaction subject with the emotion represented by this reaction group. **Arguments for `reactors`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`subject` (`Reactable!`)	The subject that was reacted to.
`users` (`ReactingUserConnection!`)	Users who have reacted to the reaction subject with the emotion represented by this reaction group.

Deprecation notice
`users` is deprecated.
Reactors can now be mannequins, bots, and organizations. Use the `reactors` field instead. Removal on 2021-10-01 UTC.

Arguments for `users`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`viewerHasReacted` (`Boolean!`)	Whether or not the authenticated user has left a reaction on the subject.

ReactorConnection

The connection type for Reactor.

Fields for `ReactorConnection`

Name	Description
`edges` (`[ReactorEdge]`)	A list of edges.
`nodes` (`[Reactor]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ReactorEdge

Represents an author of a reaction.

Fields for `ReactorEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Reactor!`)	The author of the reaction.

`reactedAt` (`DateTime!`)	The moment when the user made the reaction.

ReadyForReviewEvent

Represents a `ready_for_review` event on a given pull request.

`ReadyForReviewEvent` Implements

- `Node`
- `UniformResourceLocatable`

Fields for `ReadyForReviewEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`pullRequest` (`PullRequest!`)	PullRequest referenced by event.
`resourcePath` (`URI!`)	The HTTP path for this ready for review event.
`url` (`URI!`)	The HTTP URL for this ready for review event.

Ref

Represents a Git reference.

`Ref` Implements

- `Node`

Fields for `Ref`

Name	Description
`associatedPullRequests` (`PullRequestConnection!`)	A list of pull requests with this ref as the head ref. **Arguments for `associatedPullRequests`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `baseRefName` (`String`) The base ref name to filter the pull requests by. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`)

Returns the first *n* elements from the list.

`headRefName` (`String`)
The head ref name to filter the pull requests by.

`labels` (`[String!]`)
A list of label names to filter the pull requests by.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`IssueOrder`)
Ordering options for pull requests returned from the connection.

`states` (`[PullRequestState!]`)
A list of states to filter the pull requests by.

`branchProtectionRule` (`BranchProtectionRule`)	Branch protection rules for this ref.
`compare` (`Comparison`)	Compares the current ref as a base ref to another head ref, if the comparison can be made.

Arguments for `compare`

`headRef` (`String!`)
The head ref to compare against.

`name` (`String!`)	The ref name.
`prefix` (`String!`)	The ref's prefix, such as `refs/heads/` or `refs/tags/`.
`refUpdateRule` (`RefUpdateRule`)	Branch protection rules that are viewable by non-admins.
`repository` (`Repository!`)	The repository the ref belongs to.
`target` (`GitObject`)	The object the ref points to. Returns null when object does not exist.

RefConnection

The connection type for Ref.

Fields for `RefConnection`

Name	Description
`edges` (`[RefEdge]`)	A list of edges.
`nodes` (`[Ref]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.

`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

RefEdge

An edge in a connection.

Fields for `RefEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Ref`)	The item at the end of the edge.

RefNameConditionTarget

Parameters to be used for the ref_name condition.

Fields for `RefNameConditionTarget`

Name	Description
`exclude` (`[String!]!`)	Array of ref names or patterns to exclude. The condition will not pass if any of these patterns match.
`include` (`[String!]!`)	Array of ref names or patterns to include. One of these patterns must match for the condition to pass. Also accepts `~DEFAULT_BRANCH` to include the default branch or `~ALL` to include all branches.

RefUpdateRule

A ref update rules for a viewer.

Fields for `RefUpdateRule`

Name	Description
`allowsDeletions` (`Boolean!`)	Can this branch be deleted.
`allowsForcePushes` (`Boolean!`)	Are force pushes allowed on this branch.
`blocksCreations` (`Boolean!`)	Can matching branches be created.
`pattern` (`String!`)	Identifies the protection rule pattern.
`requiredApprovingReviewCount` (`Int`)	Number of approving reviews required to update matching branches.

`requiredStatusCheckContexts` (`[String]`)	List of required status check contexts that must pass for commits to be accepted to matching branches.
`requiresCodeOwnerReviews` (`Boolean!`)	Are reviews from code owners required to update matching branches.
`requiresConversationResolution` (`Boolean!`)	Are conversations required to be resolved before merging.
`requiresLinearHistory` (`Boolean!`)	Are merge commits prohibited from being pushed to this branch.
`requiresSignatures` (`Boolean!`)	Are commits required to be signed.
`viewerAllowedToDismissReviews` (`Boolean!`)	Is the viewer allowed to dismiss reviews.
`viewerCanPush` (`Boolean!`)	Can the viewer push to the branch.

ReferencedEvent

Represents a `referenced` event on a given `ReferencedSubject`.

`ReferencedEvent` Implements

- `Node`

Fields for `ReferencedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`commit` (`Commit`)	Identifies the commit associated with the `referenced` event.
`commitRepository` (`Repository!`)	Identifies the repository associated with the `referenced` event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`isCrossRepository` (`Boolean!`)	Reference originated in a different repository.
`isDirectReference` (`Boolean!`)	Checks if the commit message itself references the subject. Can be false in the case of a commit comment reference.
`subject` (`ReferencedSubject!`)	Object referenced by event.

Release

A release contains the content for a release.

`Release` Implements

- `Node`

- Reactable
- UniformResourceLocatable

Fields for `Release`

Name	Description
`author` (User)	The author of the release.
`createdAt` (DateTime!)	Identifies the date and time when the object was created.
`databaseId` (Int)	Identifies the primary key from the database.
`description` (String)	The description of the release.
`descriptionHTML` (HTML)	The description of this release rendered to HTML.
`isDraft` (Boolean!)	Whether or not the release is a draft.
`isLatest` (Boolean!)	Whether or not the release is the latest releast.
`isPrerelease` (Boolean!)	Whether or not the release is a prerelease.
`mentions` (UserConnection)	A list of users mentioned in the release description. **Arguments for `mentions`** `after` (String) Returns the elements in the list that come after the specified cursor. `before` (String) Returns the elements in the list that come before the specified cursor. `first` (Int) Returns the first *n* elements from the list. `last` (Int) Returns the last *n* elements from the list.
`name` (String)	The title of the release.
`publishedAt` (DateTime)	Identifies the date and time when the release was created.
`reactionGroups` ([ReactionGroup!])	A list of reactions grouped by content left on the subject.
`reactions` (ReactionConnection!)	A list of Reactions left on the Issue. **Arguments for `reactions`** `after` (String) Returns the elements in the list that come after the specified cursor.

after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`content` (`ReactionContent`)
Allows filtering Reactions by emoji.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ReactionOrder`)
Allows specifying the order in which reactions are returned.

`releaseAssets` (`ReleaseAssetConnection!`)	List of releases assets which are dependent on this release. **Arguments for `releaseAssets`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `name` (`String`) A list of names to filter the assets by.
`repository` (`Repository!`)	The repository that the release belongs to.
`resourcePath` (`URI!`)	The HTTP path for this issue.
`shortDescriptionHTML` (`HTML`)	A description of the release, rendered to HTML without any links in it. **Arguments for `shortDescriptionHTML`** `limit` (`Int`) How many characters to return. The default value is `200`.
`tag` (`Ref`)	The Git tag the release points to.
`tagCommit` (`Commit`)	The tag commit for this release.

`tagName` (`String!`)	The name of the release's Git tag.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this issue.
`viewerCanReact` (`Boolean!`)	Can user react to this subject.

ReleaseAsset

A release asset contains the content for a release asset.

`ReleaseAsset` Implements

- `Node`

Fields for `ReleaseAsset`

Name	Description
`contentType` (`String!`)	The asset's content-type.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`downloadCount` (`Int!`)	The number of times this asset was downloaded.
`downloadUrl` (`URI!`)	Identifies the URL where you can download the release asset via the browser.
`name` (`String!`)	Identifies the title of the release asset.
`release` (`Release`)	Release that the asset is associated with.
`size` (`Int!`)	The size (in bytes) of the asset.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`uploadedBy` (`User!`)	The user that performed the upload.
`url` (`URI!`)	Identifies the URL of the release asset.

ReleaseAssetConnection

The connection type for ReleaseAsset.

Fields for `ReleaseAssetConnection`

Name	Description
`edges` (`[ReleaseAssetEdge]`)	A list of edges.
`nodes` (`[ReleaseAsset]`)	A list of nodes.

`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ReleaseAssetEdge

An edge in a connection.

Fields for `ReleaseAssetEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`ReleaseAsset`)	The item at the end of the edge.

ReleaseConnection

The connection type for Release.

Fields for `ReleaseConnection`

Name	Description
`edges` (`[ReleaseEdge]`)	A list of edges.
`nodes` (`[Release]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ReleaseEdge

An edge in a connection.

Fields for `ReleaseEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Release`)	The item at the end of the edge.

RemovedFromMergeQueueEvent

Represents a `removed_from_merge_queue` event on a given pull request.

`RemovedFromMergeQueueEvent` Implements

- `Node`

Fields for `RemovedFromMergeQueueEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`beforeCommit` (`Commit`)	Identifies the before commit SHA for the `removed_from_merge_queue` event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`enqueuer` (`User`)	The user who removed this Pull Request from the merge queue.
`mergeQueue` (`MergeQueue`)	The merge queue where this pull request was removed from.
`pullRequest` (`PullRequest`)	PullRequest referenced by event.
`reason` (`String`)	The reason this pull request was removed from the queue.

RemovedFromProjectEvent

Represents a `removed_from_project` event on a given issue or pull request.

`RemovedFromProjectEvent` Implements

- `Node`

Fields for `RemovedFromProjectEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`project` (`Project`)	Project referenced by event. **Preview notice** `project` is available under the Project event details preview. During the preview period, the API may change without notice.
`projectColumnName` (`String!`)	Column name referenced by this project event.

Preview notice
`projectColumnName` is available under the Project event details preview. During the preview period, the API may change without notice.

RenamedTitleEvent

Represents a `renamed` event on a given issue or pull request.

`RenamedTitleEvent` Implements

- `Node`

Fields for `RenamedTitleEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`currentTitle` (`String!`)	Identifies the current title of the issue or pull request.
`previousTitle` (`String!`)	Identifies the previous title of the issue or pull request.
`subject` (`RenamedTitleSubject!`)	Subject that was renamed.

ReopenedEvent

Represents a `reopened` event on any `Closable`.

`ReopenedEvent` Implements

- `Node`

Fields for `ReopenedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`closable` (`Closable!`)	Object that was reopened.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`stateReason` (`IssueStateReason`)	The reason the issue state was changed to open.

RepoAccessAuditEntry

Audit log entry for a repo.access event.

`RepoAccessAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `RepoAccessAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.

`userUrl` (`URI`)	The HTTP URL for the user.
`visibility` (`RepoAccessAuditEntryVisibility`)	The visibility of the repository.

RepoAddMemberAuditEntry

Audit log entry for a repo.add_member event.

`RepoAddMemberAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `RepoAddMemberAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.
`actorLogin` (`String`)	For actions involving two users, the actor is the

`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.
`visibility` (`RepoAddMemberAuditEntryVisibility`)	The visibility of the repository.

RepoAddTopicAuditEntry

Audit log entry for a repo.add_topic event.

`RepoAddTopicAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`
- `TopicAuditEntryData`

Fields for `RepoAddTopicAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.

`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`topic` (`Topic`)	The name of the topic added to the repository.
`topicName` (`String`)	The name of the topic added to the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

RepoArchivedAuditEntry

Audit log entry for a repo.archived event.

`RepoArchivedAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `RepoArchivedAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.

`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.
`visibility` (`RepoArchivedAuditEntryVisibility`)	The visibility of the repository.

RepoChangeMergeSettingAuditEntry

Audit log entry for a repo.change_merge_setting event.

`RepoChangeMergeSettingAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `RepoChangeMergeSettingAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`isEnabled` (`Boolean`)	Whether the change was to enable (true) or disable (false) the merge type.
`mergeType` (`RepoChangeMergeSettingAuditEntryMergeType`)	The merge method affected by the change.

`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

RepoConfigDisableAnonymousGitAccessAuditEntry

Audit log entry for a repo.config.disable_anonymous_git_access event.

`RepoConfigDisableAnonymousGitAccessAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `RepoConfigDisableAnonymousGitAccessAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.

`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

RepoConfigDisableCollaboratorsOnlyAuditEntry

Audit log entry for a repo.config.disable_collaborators_only event.

`RepoConfigDisableCollaboratorsOnlyAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `RepoConfigDisableCollaboratorsOnlyAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.

`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

RepoConfigDisableContributorsOnlyAuditEntry

Audit log entry for a repo.config.disable_contributors_only event.

`RepoConfigDisableContributorsOnlyAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `RepoConfigDisableContributorsOnlyAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.

`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

RepoConfigDisableSockpuppetDisallowedAuditEntry

Audit log entry for a repo.config.disable_sockpuppet_disallowed event.

`RepoConfigDisableSockpuppetDisallowedAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `RepoConfigDisableSockpuppetDisallowedAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.

`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

RepoConfigEnableAnonymousGitAccessAuditEntry

Audit log entry for a repo.config.enable_anonymous_git_access event.

`RepoConfigEnableAnonymousGitAccessAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `RepoConfigEnableAnonymousGitAccessAuditEntry`

Name	Description
`action` (`String!`)	The action name.

`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

RepoConfigEnableCollaboratorsOnlyAuditEntry

Audit log entry for a repo.config.enable_collaborators_only event.

`RepoConfigEnableCollaboratorsOnlyAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `RepoConfigEnableCollaboratorsOnlyAuditEntry`

Name	Description

`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

RepoConfigEnableContributorsOnlyAuditEntry

Audit log entry for a repo.config.enable_contributors_only event.

`RepoConfigEnableContributorsOnlyAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `RepoConfigEnableContributorsOnlyAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

RepoConfigEnableSockpuppetDisallowedAuditEntry

Audit log entry for a repo.config.enable_sockpuppet_disallowed event.

`RepoConfigEnableSockpuppetDisallowedAuditEntry` Implements

- `AuditEntry`

- Node
- OrganizationAuditEntryData
- RepositoryAuditEntryData

Fields for `RepoConfigEnableSockpuppetDisallowedAuditEntry`

Name	Description
`action` (String!)	The action name.
`actor` (AuditEntryActor)	The user who initiated the action.
`actorIp` (String)	The IP address of the actor.
`actorLocation` (ActorLocation)	A readable representation of the actor's location.
`actorLogin` (String)	The username of the user who initiated the action.
`actorResourcePath` (URI)	The HTTP path for the actor.
`actorUrl` (URI)	The HTTP URL for the actor.
`createdAt` (PreciseDateTime!)	The time the action was initiated.
`operationType` (OperationType)	The corresponding operation type for the action.
`organization` (Organization)	The Organization associated with the Audit Entry.
`organizationName` (String)	The name of the Organization.
`organizationResourcePath` (URI)	The HTTP path for the organization.
`organizationUrl` (URI)	The HTTP URL for the organization.
`repository` (Repository)	The repository associated with the action.
`repositoryName` (String)	The name of the repository.
`repositoryResourcePath` (URI)	The HTTP path for the repository.
`repositoryUrl` (URI)	The HTTP URL for the repository.
`user` (User)	The user affected by the action.
`userLogin` (String)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (URI)	The HTTP path for the user.
`userUrl` (URI)	The HTTP URL for the user.

RepoConfigLockAnonymousGitAccessAuditEntry

Audit log entry for a repo.config.lock_anonymous_git_access event.

`RepoConfigLockAnonymousGitAccessAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `RepoConfigLockAnonymousGitAccessAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

RepoConfigUnlockAnonymousGitAccessAuditEntry

Audit log entry for a repo.config.unlock_anonymous_git_access event.

`RepoConfigUnlockAnonymousGitAccessAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `RepoConfigUnlockAnonymousGitAccessAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

RepoCreateAuditEntry

Audit log entry for a repo.create event.

`RepoCreateAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `RepoCreateAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`forkParentName` (`String`)	The name of the parent repository for this forked repository.
`forkSourceName` (`String`)	The name of the root repository for this network.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.

`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.
`visibility` (`RepoCreateAuditEntryVisibility`)	The visibility of the repository.

RepoDestroyAuditEntry

Audit log entry for a repo.destroy event.

`RepoDestroyAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `RepoDestroyAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.

`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.
`visibility` (`RepoDestroyAuditEntryVisibility`)	The visibility of the repository.

RepoRemoveMemberAuditEntry

Audit log entry for a repo.remove_member event.

`RepoRemoveMemberAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`

Fields for `RepoRemoveMemberAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.

`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.
`visibility` (`RepoRemoveMemberAuditEntryVisibility`)	The visibility of the repository.

RepoRemoveTopicAuditEntry

Audit log entry for a repo.remove_topic event.

`RepoRemoveTopicAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`
- `TopicAuditEntryData`

Fields for `RepoRemoveTopicAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.

`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`topic` (`Topic`)	The name of the topic added to the repository.
`topicName` (`String`)	The name of the topic added to the repository.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

Repository

A repository contains the content for a project.

`Repository` Implements

- `Node`
- `PackageOwner`
- `ProjectOwner`
- `ProjectV2Recent`
- `RepositoryInfo`
- `Starrable`
- `Subscribable`
- `UniformResourceLocatable`

Fields for `Repository`

Name	Description
`allowUpdateBranch` (`Boolean!`)	Whether or not a pull request head branch that is behind its base branch can always be updated even if it is not required to be up to date before merging.
`archivedAt` (`DateTime`)	Identifies the date and time when the repository was archived.
`assignableUsers` (`UserConnection!`)	A list of users that can be assigned to issues in this repository.

Arguments for `assignableUsers`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`query` (`String`)
Filters users with query on user name and login.

`autoMergeAllowed` (`Boolean!`)

Whether or not Auto-merge can be enabled on pull requests in this repository.

`branchProtectionRules` (`BranchProtectionRuleConnection!`)

A list of branch protection rules for this repository.

Arguments for `branchProtectionRules`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`codeOfConduct` (`CodeOfConduct`)

Returns the code of conduct for this repository.

`codeowners` (`RepositoryCodeowners`)

Information extracted from the repository's `CODEOWNERS` file.

Arguments for `codeowners`

`refName` (`String`)
The ref name used to return the associated `CODEOWNERS` file.

`collaborators` (`RepositoryCollaboratorConnection`)

A list of collaborators associated with the repository.

Arguments for `collaborators`

`affiliation` (`CollaboratorAffiliation`)
Collaborators affiliation level with a repository.

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`login` (`String`)
The login of one specific collaborator.

`query` (`String`)
Filters users with query on user name and login.

`commitComments` (`CommitCommentConnection!`)

A list of commit comments associated with the repository.

Arguments for `commitComments`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`contactLinks` (`[RepositoryContactLink!]`)

Returns a list of contact links associated to the repository.

`contributingGuidelines` (`ContributingGuidelines`)

Returns the contributing guidelines for this repository.

`createdAt` (`DateTime!`)

Identifies the date and time when the object was created.

`databaseId` (`Int`)

Identifies the primary key from the database.

`defaultBranchRef` (`Ref`)

The Ref associated with the repository's default branch.

branch.

`deleteBranchOnMerge` (`Boolean!`)	Whether or not branches are automatically deleted when merged in this repository.
`dependencyGraphManifests` (`DependencyGraphManifestConnection`)	A list of dependency manifests contained in the repository.

Preview notice
`dependencyGraphManifests` is available under the Access to a repository's dependency graph preview. During the preview period, the API may change without notice.

Arguments for `dependencyGraphManifests`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`dependenciesAfter` (`String`)
Cursor to paginate dependencies.

`dependenciesFirst` (`Int`)
Number of dependencies to fetch.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`withDependencies` (`Boolean`)
Flag to scope to only manifests with dependencies.

`deployKeys` (`DeployKeyConnection!`)	A list of deploy keys that are on this repository.

Arguments for `deployKeys`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)

Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`deployments` (`DeploymentConnection!`)	Deployments associated with the repository. **Arguments for `deployments`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `environments` (`[String!]`) Environments to list deployments for. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`DeploymentOrder`) Ordering options for deployments returned from the connection.
`description` (`String`)	The description of the repository.
`descriptionHTML` (`HTML!`)	The description of the repository rendered to HTML.
`discussion` (`Discussion`)	Returns a single discussion from the current repository by number. **Arguments for `discussion`** `number` (`Int!`) The number for the discussion to be returned.
`discussionCategories` (`DiscussionCategoryConnection!`)	A list of discussion categories that are available in the repository. **Arguments for `discussionCategories`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `filterByAssignable` (`Boolean`) Filter by categories that are assignable by the viewer. The default value is `false` .

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`discussionCategory` (`DiscussionCategory`)

A discussion category by slug.

Arguments for `discussionCategory`

`slug` (`String!`)
The slug of the discussion category to be returned.

`discussions` (`DiscussionConnection!`)

A list of discussions that have been opened in the repository.

Arguments for `discussions`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`answered` (`Boolean`)
Only show answered or unanswered discussions.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`categoryId` (`ID`)
Only include discussions that belong to the category with this ID.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`DiscussionOrder`)
Ordering options for discussions returned from the connection.

`states` (`[DiscussionState!]`)
A list of states to filter the discussions by.

`diskUsage` (`Int`)

The number of kilobytes this repository occupies on disk.

`environment` (`Environment`)

Returns a single active environment from the current repository by name.

Arguments for `environment`

`name` (`String!`)

The name of the environment to be returned.

`environments` (`EnvironmentConnection!`)

A list of environments that are in this repository.

Arguments for `environments`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`Environments`)
Ordering options for the environments.

`forkCount` (`Int!`)

Returns how many forks there are of this repository in the whole network.

`forkingAllowed` (`Boolean!`)

Whether this repository allows forks.

`forks` (`RepositoryConnection!`)

A list of direct forked repositories.

Arguments for `forks`

`affiliations` (`[RepositoryAffiliation]`)
Array of viewer's affiliation options for repositories returned from the connection. For example, OWNER will include only repositories that the current viewer owns.

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`hasIssuesEnabled` (`Boolean`)
If non-null, filters repositories according to whether they have issues enabled.

`isLocked` (`Boolean`)
If non-null, filters repositories according to whether they have been locked.

`last` (`Int`)

Returns the last *n* elements from the list.

`orderBy` (`RepositoryOrder`)
Ordering options for repositories returned from the connection.

`ownerAffiliations` (`[RepositoryAffiliation]`)
Array of owner's affiliation options for repositories returned from the connection. For example, OWNER will include only repositories that the organization or user being viewed owns.

`privacy` (`RepositoryPrivacy`)
If non-null, filters repositories according to privacy.

`fundingLinks` (`[FundingLink!]!`)	The funding links for this repository.
`hasDiscussionsEnabled` (`Boolean!`)	Indicates if the repository has the Discussions feature enabled.
`hasIssuesEnabled` (`Boolean!`)	Indicates if the repository has issues feature enabled.
`hasProjectsEnabled` (`Boolean!`)	Indicates if the repository has the Projects feature enabled.
`hasVulnerabilityAlertsEnabled` (`Boolean!`)	Whether vulnerability alerts are enabled for the repository.
`hasWikiEnabled` (`Boolean!`)	Indicates if the repository has wiki feature enabled.
`homepageUrl` (`URI`)	The repository's URL.
`interactionAbility` (`RepositoryInteractionAbility`)	The interaction ability settings for this repository.
`isArchived` (`Boolean!`)	Indicates if the repository is unmaintained.
`isBlankIssuesEnabled` (`Boolean!`)	Returns true if blank issue creation is allowed.
`isDisabled` (`Boolean!`)	Returns whether or not this repository disabled.
`isEmpty` (`Boolean!`)	Returns whether or not this repository is empty.
`isFork` (`Boolean!`)	Identifies if the repository is a fork.
`isInOrganization` (`Boolean!`)	Indicates if a repository is either owned by an organization, or is a private fork of an organization repository.
`isLocked` (`Boolean!`)	Indicates if the repository has been locked or not.
`isMirror` (`Boolean!`)	Identifies if the repository is a mirror.
`isPrivate` (`Boolean!`)	Identifies if the repository is private or internal.

`isSecurityPolicyEnabled` (`Boolean`)	Returns true if this repository has a security policy.
`isTemplate` (`Boolean!`)	Identifies if the repository is a template that can be used to generate new repositories.
`isUserConfigurationRepository` (`Boolean!`)	Is this repository a user configuration repository?.
`issue` (`Issue`)	Returns a single issue from the current repository by number. **Arguments for `issue`** `number` (`Int!`) The number for the issue to be returned.
`issueOrPullRequest` (`IssueOrPullRequest`)	Returns a single issue-like object from the current repository by number. **Arguments for `issueOrPullRequest`** `number` (`Int!`) The number for the issue to be returned.
`issueTemplates` (`[IssueTemplate!]`)	Returns a list of issue templates associated to the repository.
`issues` (`IssueConnection!`)	A list of issues that have been opened in the repository. **Arguments for `issues`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `filterBy` (`IssueFilters`) Filtering options for issues returned from the connection. `first` (`Int`) Returns the first *n* elements from the list. `labels` (`[String!]`) A list of label names to filter the pull requests by. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`IssueOrder`) Ordering options for issues returned from the connection. `states` (`[IssueState!]`)

`states` (`[IssueState!]`)
A list of states to filter the issues by.

`label` (`Label`)

Returns a single label by name.

Arguments for `label`

`name` (`String!`)
Label name.

`labels` (`LabelConnection`)

A list of labels associated with the repository.

Arguments for `labels`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`LabelOrder`)
Ordering options for labels returned from the connection.

`query` (`String`)
If provided, searches labels by name and description.

`languages` (`LanguageConnection`)

A list containing a breakdown of the language composition of the repository.

Arguments for `languages`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`LanguageOrder`)
Order for connection.

`latestRelease` (`Release`)	Get the latest release for the repository if one exists.
`licenseInfo` (`License`)	The license associated with the repository.
`lockReason` (`RepositoryLockReason`)	The reason the repository has been locked.
`mentionableUsers` (`UserConnection!`)	A list of Users that can be mentioned in the context of the repository. **Arguments for `mentionableUsers`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `query` (`String`) Filters users with query on user name and login.
`mergeCommitAllowed` (`Boolean!`)	Whether or not PRs are merged with a merge commit on this repository.
`mergeCommitMessage` (`MergeCommitMessage!`)	How the default commit message will be generated when merging a pull request.
`mergeCommitTitle` (`MergeCommitTitle!`)	How the default commit title will be generated when merging a pull request.
`mergeQueue` (`MergeQueue`)	The merge queue for a specified branch, otherwise the default branch if not provided. **Arguments for `mergeQueue`** `branch` (`String`) The name of the branch to get the merge queue for. Case sensitive.
`milestone` (`Milestone`)	Returns a single milestone from the current repository by number. **Arguments for `milestone`** `number` (`Int!`) The number for the milestone to be returned.

`milestones` (`MilestoneConnection`)	A list of milestones associated with the repository. **Arguments for `milestones`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`MilestoneOrder`) Ordering options for milestones. `query` (`String`) Filters milestones with a query on the title. `states` (`[MilestoneState!]`) Filter by the state of the milestones.
`mirrorUrl` (`URI`)	The repository's original mirror URL.
`name` (`String!`)	The name of the repository.
`nameWithOwner` (`String!`)	The repository's name with owner.
`object` (`GitObject`)	A Git object in the repository. **Arguments for `object`** `expression` (`String`) A Git revision expression suitable for rev-parse. `oid` (`GitObjectID`) The Git object ID.
`openGraphImageUrl` (`URI!`)	The image used to represent this repository in Open Graph data.
`owner` (`RepositoryOwner!`)	The User owner of the repository.
`packages` (`PackageConnection!`)	A list of packages under the owner. **Arguments for `packages`** `after` (`String`)

Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`names` (`[String]`)
Find packages by their names.

`orderBy` (`PackageOrder`)
Ordering of the returned packages.

`packageType` (`PackageType`)
Filter registry package by type.

`repositoryId` (`ID`)
Find packages in a repository by ID.

`parent` (`Repository`)	The repository parent, if this is a fork.
`pinnedDiscussions` (`PinnedDiscussionConnection!`)	A list of discussions that have been pinned in this repository.

Arguments for `pinnedDiscussions`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`pinnedIssues` (`PinnedIssueConnection`)	A list of pinned issues for this repository.

Arguments for `pinnedIssues`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)

Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`primaryLanguage` (`Language`)	The primary language of the repository's code.
`project` (`Project`)	Find project by number. **Arguments for `project`** `number` (`Int!`) The project number to find.
`projectV2` (`ProjectV2`)	Finds and returns the Project according to the provided Project number. **Arguments for `projectV2`** `number` (`Int!`) The Project number.
`projects` (`ProjectConnection!`)	A list of projects under the owner. **Arguments for `projects`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`ProjectOrder`) Ordering options for projects returned from the connection. `search` (`String`) Query to search projects by, currently only searching by name. `states` (`[ProjectState!]`) A list of states to filter the projects by.
`projectsResourcePath` (`URI!`)	The HTTP path listing the repository's projects.
`projectsUrl` (`URI!`)	The HTTP URL listing the repository's projects.

`projectsV2` (`ProjectV2Connection!`)

List of projects linked to this repository.

Arguments for `projectsV2`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ProjectV2Order`)
How to order the returned projects.

`query` (`String`)
A project to search for linked to the repo.

`pullRequest` (`PullRequest`)

Returns a single pull request from the current repository by number.

Arguments for `pullRequest`

`number` (`Int!`)
The number for the pull request to be returned.

`pullRequestTemplates` (`[PullRequestTemplate!]`)

Returns a list of pull request templates associated to the repository.

`pullRequests` (`PullRequestConnection!`)

A list of pull requests that have been opened in the repository.

Arguments for `pullRequests`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`baseRefName` (`String`)
The base ref name to filter the pull requests by.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`headRefName` (`String`)
The head ref name to filter the pull requests by.

requests by.

`labels` (`[String!]`)
A list of label names to filter the pull requests by.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`IssueOrder`)
Ordering options for pull requests returned from the connection.

`states` (`[PullRequestState!]`)
A list of states to filter the pull requests by.

`pushedAt` (`DateTime`)	Identifies the date and time when the repository was last pushed to.
`rebaseMergeAllowed` (`Boolean!`)	Whether or not rebase-merging is enabled on this repository.
`recentProjects` (`ProjectV2Connection!`)	Recent projects that this user has modified in the context of the owner.

Arguments for `recentProjects`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`ref` (`Ref`)	Fetch a given ref from the repository.

Arguments for `ref`

`qualifiedName` (`String!`)
The ref to retrieve. Fully qualified matches are checked in order (`refs/heads/master`)

before falling back onto checks for short name matches (`master`).

`refs` (`RefConnection`)	Fetch a list of refs from the repository.

Arguments for `refs`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`direction` (`OrderDirection`)
DEPRECATED: use orderBy. The ordering direction.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`RefOrder`)
Ordering options for refs returned from the connection.

`query` (`String`)
Filters refs with query on name.

`refPrefix` (`String!`)
A ref name prefix like `refs/heads/`, `refs/tags/`, etc.

`release` (`Release`)

Lookup a single release given various criteria.

Arguments for `release`

`tagName` (`String!`)
The name of the Tag the Release was created from.

`releases` (`ReleaseConnection!`)

List of releases which are dependent on this repository.

Arguments for `releases`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ReleaseOrder`)
Order for connection.

`repositoryTopics` (`RepositoryTopicConnection!`)	A list of applied repository-topic associations for this repository. **Arguments for `repositoryTopics`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`resourcePath` (`URI!`)	The HTTP path for this repository.
`ruleset` (`RepositoryRuleset`)	Returns a single ruleset from the current repository by ID. **Arguments for `ruleset`** `databaseId` (`Int!`) The ID of the ruleset to be returned. `includeParents` (`Boolean`) Include rulesets configured at higher levels that apply to this repository. The default value is `true`.
`rulesets` (`RepositoryRulesetConnection`)	A list of rulesets for this repository. **Arguments for `rulesets`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `includeParents` (`Boolean`) Return rulesets configured at higher levels that apply to this repository. The default value is `true`. `last` (`Int`) Returns the last *n* elements from the list.

`securityPolicyUrl` (`URI`)	The security policy URL.
`shortDescriptionHTML` (`HTML!`)	A description of the repository, rendered to HTML without any links in it. **Arguments for `shortDescriptionHTML`** `limit` (`Int`) How many characters to return. The default value is `200`.
`squashMergeAllowed` (`Boolean!`)	Whether or not squash-merging is enabled on this repository.
`squashMergeCommitMessage` (`SquashMergeCommitMessage!`)	How the default commit message will be generated when squash merging a pull request.
`squashMergeCommitTitle` (`SquashMergeCommitTitle!`)	How the default commit title will be generated when squash merging a pull request.
`squashPrTitleUsedAsDefault` (`Boolean!`)	Whether a squash merge commit can use the pull request title as default. **Deprecation notice** `squashPrTitleUsedAsDefault` is deprecated. `squashPrTitleUsedAsDefault` will be removed. Use `Repository.squashMergeCommitTitle` instead. Removal on 2023-04-01 UTC.
`sshUrl` (`GitSSHRemote!`)	The SSH URL to clone this repository.
`stargazerCount` (`Int!`)	Returns a count of how many stargazers there are on this object.
`stargazers` (`StargazerConnection!`)	A list of users who have starred this starrable. **Arguments for `stargazers`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`StarOrder`) Order for connection.

`submodules` (`SubmoduleConnection!`)	Returns a list of all submodules in this repository parsed from the .gitmodules file as of the default branch's HEAD commit. **Arguments for `submodules`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`tempCloneToken` (`String`)	Temporary authentication token for cloning this repository.
`templateRepository` (`Repository`)	The repository from which this repository was generated, if any.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this repository.
`usesCustomOpenGraphImage` (`Boolean!`)	Whether this repository has a custom image to use with Open Graph as opposed to being represented by the owner's avatar.
`viewerCanAdminister` (`Boolean!`)	Indicates whether the viewer has admin permissions on this repository.
`viewerCanCreateProjects` (`Boolean!`)	Can the current viewer create new projects on this owner.
`viewerCanSubscribe` (`Boolean!`)	Check if the viewer is able to change their subscription status for the repository.
`viewerCanUpdateTopics` (`Boolean!`)	Indicates whether the viewer can update the topics of this repository.
`viewerDefaultCommitEmail` (`String`)	The last commit email for the viewer.
`viewerDefaultMergeMethod` (`PullRequestMergeMethod!`)	The last used merge method by the viewer or the default for the repository.
`viewerHasStarred` (`Boolean!`)	Returns a boolean indicating whether the viewing user has starred this starrable.
`viewerPermission` (`RepositoryPermission`)	The users permission level on the repository. Will return null if authenticated as an GitHub App.
`viewerPossibleCommitEmails` (`[String!]`)	A list of emails this viewer can commit with.

`viewerSubscription` (`SubscriptionState`)	Identifies if the viewer is watching, not watching, or ignoring the subscribable entity.
`visibility` (`RepositoryVisibility!`)	Indicates the repository's visibility level.
`vulnerabilityAlert` (`RepositoryVulnerabilityAlert`)	Returns a single vulnerability alert from the current repository by number. **Arguments for `vulnerabilityAlert`** `number` (`Int!`) The number for the vulnerability alert to be returned.
`vulnerabilityAlerts` (`RepositoryVulnerabilityAlertConnection`)	A list of vulnerability alerts that are on this repository. **Arguments for `vulnerabilityAlerts`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `dependencyScopes` (`[RepositoryVulnerabilityAlertDependencyScope!]`) Filter by the scope of the alert's dependency. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `states` (`[RepositoryVulnerabilityAlertState!]`) Filter by the state of the alert.
`watchers` (`UserConnection!`)	A list of users watching the repository. **Arguments for `watchers`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`)

	`last` (`Int`) Returns the last *n* elements from the list.
`webCommitSignoffRequired` (`Boolean!`)	Whether contributors are required to sign off on web-based commits in this repository.

RepositoryCodeowners

Information extracted from a repository's `CODEOWNERS` file.

Fields for `RepositoryCodeowners`

Name	Description
`errors` (`[RepositoryCodeownersError!]!`)	Any problems that were encountered while parsing the `CODEOWNERS` file.

RepositoryCodeownersError

An error in a `CODEOWNERS` file.

Fields for `RepositoryCodeownersError`

Name	Description
`column` (`Int!`)	The column number where the error occurs.
`kind` (`String!`)	A short string describing the type of error.
`line` (`Int!`)	The line number where the error occurs.
`message` (`String!`)	A complete description of the error, combining information from other fields.
`path` (`String!`)	The path to the file when the error occurs.
`source` (`String!`)	The content of the line where the error occurs.
`suggestion` (`String`)	A suggestion of how to fix the error.

RepositoryCollaboratorConnection

The connection type for User.

Fields for `RepositoryCollaboratorConnection`

Name	Description
`edges` (`[RepositoryCollaboratorEdge]`)	A list of edges.
`nodes` (`[User]`)	A list of nodes.

`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

RepositoryCollaboratorEdge

Represents a user who is a collaborator of a repository.

Fields for `RepositoryCollaboratorEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`permission` (`RepositoryPermission!`)	The permission the user has on the repository.
`permissionSources` (`[PermissionSource!]`)	A list of sources for the user's access to the repository.

RepositoryConnection

A list of repositories owned by the subject.

Fields for `RepositoryConnection`

Name	Description
`edges` (`[RepositoryEdge]`)	A list of edges.
`nodes` (`[Repository]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.
`totalDiskUsage` (`Int!`)	The total size in kilobytes of all repositories in the connection.

RepositoryContactLink

A repository contact link.

Fields for `RepositoryContactLink`

Name	Description
`about` (`String!`)	The contact link purpose.
`name` (`String!`)	The contact link name.
`url` (`URI!`)	The contact link URL.

RepositoryEdge

An edge in a connection.

Fields for `RepositoryEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Repository`)	The item at the end of the edge.

RepositoryIdConditionTarget

Parameters to be used for the repository_id condition.

Fields for `RepositoryIdConditionTarget`

Name	Description
`repositoryIds` (`[ID!]!`)	One of these repo IDs must match the repo.

RepositoryInteractionAbility

Repository interaction limit that applies to this object.

Fields for `RepositoryInteractionAbility`

Name	Description
`expiresAt` (`DateTime`)	The time the currently active limit expires.
`limit` (`RepositoryInteractionLimit!`)	The current limit that is enabled on this object.
`origin` (`RepositoryInteractionLimitOrigin!`)	The origin of the currently active interaction limit.

RepositoryInvitation

An invitation for a user to be added to a repository.

`RepositoryInvitation` Implements

- `Node`

Fields for `RepositoryInvitation`

Name	Description
`email` (`String`)	The email address that received the invitation.
`invitee` (`User`)	The user who received the invitation.

`inviter` (`User!`)	The user who created the invitation.
`permalink` (`URI!`)	The permalink for this repository invitation.
`permission` (`RepositoryPermission!`)	The permission granted on this repository by this invitation.
`repository` (`RepositoryInfo`)	The Repository the user is invited to.

RepositoryInvitationConnection

A list of repository invitations.

Fields for `RepositoryInvitationConnection`

Name	Description
`edges` (`[RepositoryInvitationEdge]`)	A list of edges.
`nodes` (`[RepositoryInvitation]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

RepositoryInvitationEdge

An edge in a connection.

Fields for `RepositoryInvitationEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`RepositoryInvitation`)	The item at the end of the edge.

RepositoryMigration

A GitHub Enterprise Importer (GEI) repository migration.

`RepositoryMigration` Implements

- `Migration`
- `Node`

Fields for `RepositoryMigration`

Name	Description
`continueOnError` (`Boolean!`)	The migration flag to continue on error.

`continueOnError` (`Boolean!`)	The migration flag to continue on error.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`String`)	Identifies the primary key from the database.
`failureReason` (`String`)	The reason the migration failed.
`migrationLogUrl` (`URI`)	The URL for the migration log (expires 1 day after migration completes).
`migrationSource` (`MigrationSource!`)	The migration source.
`repositoryName` (`String!`)	The target repository name.
`sourceUrl` (`URI!`)	The migration source URL, for example `https://github.com` or `https://monalisa.ghe.com`.
`state` (`MigrationState!`)	The migration state.
`warningsCount` (`Int!`)	The number of warnings encountered for this migration. To review the warnings, check the Migration Log.

RepositoryMigrationConnection

The connection type for RepositoryMigration.

Fields for `RepositoryMigrationConnection`

Name	Description
`edges` (`[RepositoryMigrationEdge]`)	A list of edges.
`nodes` (`[RepositoryMigration]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

RepositoryMigrationEdge

Represents a repository migration.

Fields for `RepositoryMigrationEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`RepositoryMigration`)	The item at the end of the edge.

RepositoryNameConditionTarget

Parameters to be used for the repository_name condition.

Fields for `RepositoryNameConditionTarget`

Name	Description
`exclude` (`[String!]!`)	Array of repository names or patterns to exclude. The condition will not pass if any of these patterns match.
`include` (`[String!]!`)	Array of repository names or patterns to include. One of these patterns must match for the condition to pass. Also accepts `~ALL` to include all repositories.
`protected` (`Boolean!`)	Target changes that match these patterns will be prevented except by those with bypass permissions.

RepositoryRule

A repository rule.

`RepositoryRule` Implements

- `Node`

Fields for `RepositoryRule`

Name	Description
`parameters` (`RuleParameters`)	The parameters for this rule.
`repositoryRuleset` (`RepositoryRuleset`)	The repository ruleset associated with this rule configuration.
`type` (`RepositoryRuleType!`)	The type of rule.

RepositoryRuleConditions

Set of conditions that determine if a ruleset will evaluate.

Fields for `RepositoryRuleConditions`

Name	Description
`refName` (`RefNameConditionTarget`)	Configuration for the ref_name condition.
`repositoryId` (`RepositoryIdConditionTarget`)	Configuration for the repository_id condition.
`repositoryName` (`RepositoryNameConditionTarget`)	Configuration for the repository_name condition.

RepositoryRuleConnection

The connection type for RepositoryRule.

Fields for `RepositoryRuleConnection`

Name	Description
`edges` (`[RepositoryRuleEdge]`)	A list of edges.
`nodes` (`[RepositoryRule]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

RepositoryRuleEdge

An edge in a connection.

Fields for `RepositoryRuleEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`RepositoryRule`)	The item at the end of the edge.

RepositoryRuleset

A repository ruleset.

`RepositoryRuleset` Implements

- `Node`

Fields for `RepositoryRuleset`

Name	Description
`bypassActors` (`RepositoryRulesetBypassActorConnection`)	The actors that can bypass this ruleset. **Arguments for `bypassActors`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.

`conditions` (`RepositoryRuleConditions!`)	The set of conditions that must evaluate to true for this ruleset to apply.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`enforcement` (`RuleEnforcement!`)	The enforcement level of this ruleset.
`name` (`String!`)	Name of the ruleset.
`rules` (`RepositoryRuleConnection`)	List of rules. **Arguments for `rules`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `type` (`RepositoryRuleType`) The type of rule.
`source` (`RuleSource!`)	Source of ruleset.
`target` (`RepositoryRulesetTarget`)	Target of the ruleset.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

RepositoryRulesetBypassActor

A team or app that has the ability to bypass a rules defined on a ruleset.

`RepositoryRulesetBypassActor` Implements

- `Node`

Fields for `RepositoryRulesetBypassActor`

Name	Description
`actor` (`BypassActor`)	The actor that can bypass rules.

`bypassMode` (`RepositoryRulesetBypassActorBypassMode`)	The mode for the bypass actor.
`organizationAdmin` (`Boolean!`)	This actor represents the ability for an organization admin to bypass.
`repositoryRoleDatabaseId` (`Int`)	If the actor is a repository role, the repository role's ID that can bypass.
`repositoryRoleName` (`String`)	If the actor is a repository role, the repository role's name that can bypass.
`repositoryRuleset` (`RepositoryRuleset`)	Identifies the ruleset associated with the allowed actor.

RepositoryRulesetBypassActorConnection

The connection type for RepositoryRulesetBypassActor.

Fields for `RepositoryRulesetBypassActorConnection`

Name	Description
`edges` (`[RepositoryRulesetBypassActorEdge]`)	A list of edges.
`nodes` (`[RepositoryRulesetBypassActor]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

RepositoryRulesetBypassActorEdge

An edge in a connection.

Fields for `RepositoryRulesetBypassActorEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`RepositoryRulesetBypassActor`)	The item at the end of the edge.

RepositoryRulesetConnection

The connection type for RepositoryRuleset.

Fields for `RepositoryRulesetConnection`

Name	Description
`edges` (`[RepositoryRulesetEdge]`)	A list of edges.
`nodes` (`[RepositoryRuleset]`)	A list of nodes.

`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

RepositoryRulesetEdge

An edge in a connection.

Fields for `RepositoryRulesetEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`RepositoryRuleset`)	The item at the end of the edge.

RepositoryTopic

A repository-topic connects a repository to a topic.

`RepositoryTopic` Implements

- `Node`
- `UniformResourceLocatable`

Fields for `RepositoryTopic`

Name	Description
`resourcePath` (`URI!`)	The HTTP path for this repository-topic.
`topic` (`Topic!`)	The topic.
`url` (`URI!`)	The HTTP URL for this repository-topic.

RepositoryTopicConnection

The connection type for RepositoryTopic.

Fields for `RepositoryTopicConnection`

Name	Description
`edges` (`[RepositoryTopicEdge]`)	A list of edges.
`nodes` (`[RepositoryTopic]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

RepositoryTopicEdge

An edge in a connection.

Fields for `RepositoryTopicEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`RepositoryTopic`)	The item at the end of the edge.

RepositoryVisibilityChangeDisableAuditEntry

Audit log entry for a repository_visibility_change.disable event.

`RepositoryVisibilityChangeDisableAuditEntry` Implements

- `AuditEntry`
- `EnterpriseAuditEntryData`
- `Node`
- `OrganizationAuditEntryData`

Fields for `RepositoryVisibilityChangeDisableAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`enterpriseResourcePath` (`URI`)	The HTTP path for this enterprise.
`enterpriseSlug` (`String`)	The slug of the enterprise.
`enterpriseUrl` (`URI`)	The HTTP URL for this enterprise.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit

	Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

RepositoryVisibilityChangeEnableAuditEntry

Audit log entry for a repository_visibility_change.enable event.

`RepositoryVisibilityChangeEnableAuditEntry` Implements

- `AuditEntry`
- `EnterpriseAuditEntryData`
- `Node`
- `OrganizationAuditEntryData`

Fields for `RepositoryVisibilityChangeEnableAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`enterpriseResourcePath` (`URI`)	The HTTP path for this enterprise.
`enterpriseSlug` (`String`)	The slug of the enterprise.
`enterpriseUrl` (`URI`)	The HTTP URL for this enterprise.
`operationType` (`OperationType`)	The corresponding operation type for the action.

`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

RepositoryVulnerabilityAlert

A Dependabot alert for a repository with a dependency affected by a security vulnerability.

`RepositoryVulnerabilityAlert` Implements

- `Node`
- `RepositoryNode`

Fields for `RepositoryVulnerabilityAlert`

Name	Description
`autoDismissedAt` (`DateTime`)	When was the alert auto-dismissed?.
`createdAt` (`DateTime!`)	When was the alert created?.
`dependabotUpdate` (`DependabotUpdate`)	The associated Dependabot update.
`dependencyScope` (`RepositoryVulnerabilityAlertDependencyScope`)	The scope of an alert's dependency.
`dismissComment` (`String`)	Comment explaining the reason the alert was dismissed.
`dismissReason` (`String`)	The reason the alert was dismissed.
`dismissedAt` (`DateTime`)	When was the alert dismissed?.
`dismisser` (`User`)	The user who dismissed the alert.
`fixedAt` (`DateTime`)	When was the alert fixed?.
`number` (`Int!`)	Identifies the alert number.
`repository` (`Repository!`)	The associated repository.
`securityAdvisory` (`SecurityAdvisory`)	The associated security advisory.

`securityVulnerability` (`SecurityVulnerability`)	The associated security vulnerability.
`state` (`RepositoryVulnerabilityAlertState!`)	Identifies the state of the alert.
`vulnerableManifestFilename` (`String!`)	The vulnerable manifest filename.
`vulnerableManifestPath` (`String!`)	The vulnerable manifest path.
`vulnerableRequirements` (`String`)	The vulnerable requirements.

RepositoryVulnerabilityAlertConnection

The connection type for RepositoryVulnerabilityAlert.

Fields for `RepositoryVulnerabilityAlertConnection`

Name	Description
`edges` (`[RepositoryVulnerabilityAlertEdge]`)	A list of edges.
`nodes` (`[RepositoryVulnerabilityAlert]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

RepositoryVulnerabilityAlertEdge

An edge in a connection.

Fields for `RepositoryVulnerabilityAlertEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`RepositoryVulnerabilityAlert`)	The item at the end of the edge.

RequestedReviewerConnection

The connection type for RequestedReviewer.

Fields for `RequestedReviewerConnection`

Name	Description
`edges` (`[RequestedReviewerEdge]`)	A list of edges.
`nodes` (`[RequestedReviewer]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.

`totalCount` (`Int!`)	Identifies the total count of items in the connection.

RequestedReviewerEdge

An edge in a connection.

Fields for `RequestedReviewerEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`RequestedReviewer`)	The item at the end of the edge.

RequiredDeploymentsParameters

Choose which environments must be successfully deployed to before refs can be merged into a branch that matches this rule.

Fields for `RequiredDeploymentsParameters`

Name	Description
`requiredDeploymentEnvironments` (`[String!]!`)	The environments that must be successfully deployed to before branches can be merged.

RequiredStatusCheckDescription

Represents a required status check for a protected branch, but not any specific run of that check.

Fields for `RequiredStatusCheckDescription`

Name	Description
`app` (`App`)	The App that must provide this status in order for it to be accepted.
`context` (`String!`)	The name of this status.

RequiredStatusChecksParameters

Choose which status checks must pass before branches can be merged into a branch that matches this rule. When enabled, commits must first be pushed to another branch, then merged or pushed directly to a ref that matches this rule after status checks have passed.

Fields for `RequiredStatusChecksParameters`

Name	Description

`requiredStatusChecks` (`[StatusCheckConfiguration!]!`)	Status checks that are required.
`strictRequiredStatusChecksPolicy` (`Boolean!`)	Whether pull requests targeting a matching branch must be tested with the latest code. This setting will not take effect unless at least one status check is enabled.

RestrictedContribution

Represents a private contribution a user made on GitHub.

`RestrictedContribution` Implements

- `Contribution`

Fields for `RestrictedContribution`

Name	Description
`isRestricted` (`Boolean!`)	Whether this contribution is associated with a record you do not have access to. For example, your own 'first issue' contribution may have been made on a repository you can no longer access.
`occurredAt` (`DateTime!`)	When this contribution was made.
`resourcePath` (`URI!`)	The HTTP path for this contribution.
`url` (`URI!`)	The HTTP URL for this contribution.
`user` (`User!`)	The user who made this contribution.

ReviewDismissalAllowance

A user, team, or app who has the ability to dismiss a review on a protected branch.

`ReviewDismissalAllowance` Implements

- `Node`

Fields for `ReviewDismissalAllowance`

Name	Description
`actor` (`ReviewDismissalAllowanceActor`)	The actor that can dismiss.
`branchProtectionRule` (`BranchProtectionRule`)	Identifies the branch protection rule associated with the allowed user, team, or app.

ReviewDismissalAllowanceConnection

The connection type for ReviewDismissalAllowance.

Fields for `ReviewDismissalAllowanceConnection`

Name	Description
`edges` (`[ReviewDismissalAllowanceEdge]`)	A list of edges.
`nodes` (`[ReviewDismissalAllowance]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ReviewDismissalAllowanceEdge

An edge in a connection.

Fields for `ReviewDismissalAllowanceEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`ReviewDismissalAllowance`)	The item at the end of the edge.

ReviewDismissedEvent

Represents a `review_dismissed` event on a given issue or pull request.

`ReviewDismissedEvent` Implements

- `Node`
- `UniformResourceLocatable`

Fields for `ReviewDismissedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`dismissalMessage` (`String`)	Identifies the optional message associated with the `review_dismissed` event.
`dismissalMessageHTML` (`String`)	Identifies the optional message associated with the event, rendered to HTML.
`previousReviewState` (`PullRequestReviewState!`)	Identifies the previous state of the review with the `review_dismissed` event.

`pullRequest` (`PullRequest!`)	PullRequest referenced by event.
`pullRequestCommit` (`PullRequestCommit`)	Identifies the commit which caused the review to become stale.
`resourcePath` (`URI!`)	The HTTP path for this review dismissed event.
`review` (`PullRequestReview`)	Identifies the review associated with the `review_dismissed` event.
`url` (`URI!`)	The HTTP URL for this review dismissed event.

ReviewRequest

A request for a user to review a pull request.

`ReviewRequest` Implements

- `Node`

Fields for `ReviewRequest`

Name	Description
`asCodeOwner` (`Boolean!`)	Whether this request was created for a code owner.
`databaseId` (`Int`)	Identifies the primary key from the database.
`pullRequest` (`PullRequest!`)	Identifies the pull request associated with this review request.
`requestedReviewer` (`RequestedReviewer`)	The reviewer that is requested.

ReviewRequestConnection

The connection type for ReviewRequest.

Fields for `ReviewRequestConnection`

Name	Description
`edges` (`[ReviewRequestEdge]`)	A list of edges.
`nodes` (`[ReviewRequest]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

ReviewRequestEdge

An edge in a connection.

Fields for `ReviewRequestEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`ReviewRequest`)	The item at the end of the edge.

ReviewRequestRemovedEvent

Represents an `review_request_removed` event on a given pull request.

`ReviewRequestRemovedEvent` Implements

- `Node`

Fields for `ReviewRequestRemovedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`pullRequest` (`PullRequest!`)	PullRequest referenced by event.
`requestedReviewer` (`RequestedReviewer`)	Identifies the reviewer whose review request was removed.

ReviewRequestedEvent

Represents an `review_requested` event on a given pull request.

`ReviewRequestedEvent` Implements

- `Node`

Fields for `ReviewRequestedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`pullRequest` (`PullRequest!`)	PullRequest referenced by event.
`requestedReviewer` (`RequestedReviewer`)	Identifies the reviewer whose review was requested.

ReviewStatusHovercardContext

A hovercard context with a message describing the current code review state of the pull request.

`ReviewStatusHovercardContext` Implements

- `HovercardContext`

Fields for `ReviewStatusHovercardContext`

Name	Description
`message` (`String!`)	A string describing this context.
`octicon` (`String!`)	An octicon to accompany this context.
`reviewDecision` (`PullRequestReviewDecision`)	The current status of the pull request with respect to code review.

SavedReply

A Saved Reply is text a user can use to reply quickly.

`SavedReply` Implements

- `Node`

Fields for `SavedReply`

Name	Description
`body` (`String!`)	The body of the saved reply.
`bodyHTML` (`HTML!`)	The saved reply body rendered to HTML.
`databaseId` (`Int`)	Identifies the primary key from the database.
`title` (`String!`)	The title of the saved reply.
`user` (`Actor`)	The user that saved this reply.

SavedReplyConnection

The connection type for SavedReply.

Fields for `SavedReplyConnection`

Name	Description
`edges` (`[SavedReplyEdge]`)	A list of edges.
`nodes` (`[SavedReply]`)	A list of nodes.

`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

SavedReplyEdge

An edge in a connection.

Fields for `SavedReplyEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`SavedReply`)	The item at the end of the edge.

SearchResultItemConnection

A list of results that matched against a search query. Regardless of the number of matches, a maximum of 1,000 results will be available across all types, potentially split across many pages.

Fields for `SearchResultItemConnection`

Name	Description
`codeCount` (`Int!`)	The total number of pieces of code that matched the search query. Regardless of the total number of matches, a maximum of 1,000 results will be available across all types.
`discussionCount` (`Int!`)	The total number of discussions that matched the search query. Regardless of the total number of matches, a maximum of 1,000 results will be available across all types.
`edges` (`[SearchResultItemEdge]`)	A list of edges.
`issueCount` (`Int!`)	The total number of issues that matched the search query. Regardless of the total number of matches, a maximum of 1,000 results will be available across all types.
`nodes` (`[SearchResultItem]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`repositoryCount` (`Int!`)	The total number of repositories that matched the search query. Regardless of the total number of matches, a maximum of 1,000 results will be available across all types.
`userCount` (`Int!`)	The total number of users that matched the search query. Regardless of the total number of matches, a maximum of 1,000 results will be available across all types.

`wikiCount` (`Int!`)	The total number of wiki pages that matched the search query. Regardless of the total number of matches, a maximum of 1,000 results will be available across all types.

SearchResultItemEdge

An edge in a connection.

Fields for `SearchResultItemEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`SearchResultItem`)	The item at the end of the edge.
`textMatches` (`[TextMatch]`)	Text matches on the result found.

SecurityAdvisory

A GitHub Security Advisory.

`SecurityAdvisory` Implements

- `Node`

Fields for `SecurityAdvisory`

Name	Description
`classification` (`SecurityAdvisoryClassification!`)	The classification of the advisory.
`cvss` (`CVSS!`)	The CVSS associated with this advisory.
`cwes` (`CWEConnection!`)	CWEs associated with this Advisory. **Arguments for `cwes`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`databaseId` (`Int`)	Identifies the primary key from the database.

`description` (`String!`)	This is a long plaintext description of the advisory.
`ghsaId` (`String!`)	The GitHub Security Advisory ID.
`identifiers` (`[SecurityAdvisoryIdentifier!]!`)	A list of identifiers for this advisory.
`notificationsPermalink` (`URI`)	The permalink for the advisory's dependabot alerts page.
`origin` (`String!`)	The organization that originated the advisory.
`permalink` (`URI`)	The permalink for the advisory.
`publishedAt` (`DateTime!`)	When the advisory was published.
`references` (`[SecurityAdvisoryReference!]!`)	A list of references for this advisory.
`severity` (`SecurityAdvisorySeverity!`)	The severity of the advisory.
`summary` (`String!`)	A short plaintext summary of the advisory.
`updatedAt` (`DateTime!`)	When the advisory was last updated.
`vulnerabilities` (`SecurityVulnerabilityConnection!`)	Vulnerabilities associated with this Advisory.

Arguments for `vulnerabilities`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`classifications` (`[SecurityAdvisoryClassification!]`)
A list of advisory classifications to filter vulnerabilities by.

`ecosystem` (`SecurityAdvisoryEcosystem`)
An ecosystem to filter vulnerabilities by.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`SecurityVulnerabilityOrder`)
Ordering options for the returned topics.

`package` (`String`)
A package name to filter vulnerabilities by.

`severities` (`[SecurityAdvisorySeverity!]`)
A list of severities to filter vulnerabilities by.

`withdrawnAt` (`DateTime`)	When the advisory was withdrawn, if it has been withdrawn.

SecurityAdvisoryConnection

The connection type for SecurityAdvisory.

Fields for `SecurityAdvisoryConnection`

Name	Description
`edges` (`[SecurityAdvisoryEdge]`)	A list of edges.
`nodes` (`[SecurityAdvisory]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

SecurityAdvisoryEdge

An edge in a connection.

Fields for `SecurityAdvisoryEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`SecurityAdvisory`)	The item at the end of the edge.

SecurityAdvisoryIdentifier

A GitHub Security Advisory Identifier.

Fields for `SecurityAdvisoryIdentifier`

Name	Description
`type` (`String!`)	The identifier type, e.g. GHSA, CVE.
`value` (`String!`)	The identifier.

SecurityAdvisoryPackage

An individual package.

Fields for `SecurityAdvisoryPackage`

Name	Description
`ecosystem` (`SecurityAdvisoryEcosystem!`)	The ecosystem the package belongs to, e.g. RUBYGEMS, NPM.
`name` (`String!`)	The package name.

SecurityAdvisoryPackageVersion

An individual package version.

Fields for `SecurityAdvisoryPackageVersion`

Name	Description
`identifier` (`String!`)	The package name or version.

SecurityAdvisoryReference

A GitHub Security Advisory Reference.

Fields for `SecurityAdvisoryReference`

Name	Description
`url` (`URI!`)	A publicly accessible reference.

SecurityVulnerability

An individual vulnerability within an Advisory.

Fields for `SecurityVulnerability`

Name	Description
`advisory` (`SecurityAdvisory!`)	The Advisory associated with this Vulnerability.
`firstPatchedVersion` (`SecurityAdvisoryPackageVersion`)	The first version containing a fix for the vulnerability.
`package` (`SecurityAdvisoryPackage!`)	A description of the vulnerable package.
`severity` (`SecurityAdvisorySeverity!`)	The severity of the vulnerability within this package.
`updatedAt` (`DateTime!`)	When the vulnerability was last updated.
`vulnerableVersionRange` (`String!`)	A string that describes the vulnerable package versions. This string follows a basic syntax with a few forms. • `= 0.2.0` denotes a single vulnerable version. • `<= 1.0.8` denotes a version range up to and including the specified version • `< 0.1.11` denotes a version range up to, but

- `< 0.1.11` denotes a version range up to, but excluding, the specified version
- `>= 4.3.0, < 4.3.5` denotes a version range with a known minimum and maximum version.
- `>= 0.0.1` denotes a version range with a known minimum, but no known maximum.

SecurityVulnerabilityConnection

The connection type for SecurityVulnerability.

Fields for `SecurityVulnerabilityConnection`

Name	Description
`edges` (`[SecurityVulnerabilityEdge]`)	A list of edges.
`nodes` (`[SecurityVulnerability]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

SecurityVulnerabilityEdge

An edge in a connection.

Fields for `SecurityVulnerabilityEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`SecurityVulnerability`)	The item at the end of the edge.

SmimeSignature

Represents an S/MIME signature on a Commit or Tag.

`SmimeSignature` Implements

- `GitSignature`

Fields for `SmimeSignature`

Name	Description
`email` (`String!`)	Email used to sign this object.
`isValid` (`Boolean!`)	True if the signature is valid and verified by GitHub.

`payload` (`String!`)	Payload for GPG signing object. Raw ODB object without the signature header.
`signature` (`String!`)	ASCII-armored signature header from object.
`signer` (`User`)	GitHub user corresponding to the email signing this commit.
`state` (`GitSignatureState!`)	The state of this signature. `VALID` if signature is valid and verified by GitHub, otherwise represents reason why signature is considered invalid.
`wasSignedByGitHub` (`Boolean!`)	True if the signature was made with GitHub's signing key.

SocialAccount

Social media profile associated with a user.

Fields for `SocialAccount`

Name	Description
`displayName` (`String!`)	Name of the social media account as it appears on the profile.
`provider` (`SocialAccountProvider!`)	Software or company that hosts the social media account.
`url` (`URI!`)	URL of the social media account.

SocialAccountConnection

The connection type for SocialAccount.

Fields for `SocialAccountConnection`

Name	Description
`edges` (`[SocialAccountEdge]`)	A list of edges.
`nodes` (`[SocialAccount]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

SocialAccountEdge

An edge in a connection.

Fields for `SocialAccountEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`SocialAccount`)	The item at the end of the edge.

SponsorConnection

The connection type for Sponsor.

Fields for `SponsorConnection`

Name	Description
`edges` (`[SponsorEdge]`)	A list of edges.
`nodes` (`[Sponsor]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

SponsorEdge

Represents a user or organization who is sponsoring someone in GitHub Sponsors.

Fields for `SponsorEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Sponsor`)	The item at the end of the edge.

SponsorableItemConnection

The connection type for SponsorableItem.

Fields for `SponsorableItemConnection`

Name	Description
`edges` (`[SponsorableItemEdge]`)	A list of edges.
`nodes` (`[SponsorableItem]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

SponsorableItemEdge

An edge in a connection.

Fields for `SponsorableItemEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`SponsorableItem`)	The item at the end of the edge.

SponsorsActivity

An event related to sponsorship activity.

`SponsorsActivity` Implements

- `Node`

Fields for `SponsorsActivity`

Name	Description
`action` (`SponsorsActivityAction!`)	What action this activity indicates took place.
`currentPrivacyLevel` (`SponsorshipPrivacy`)	The sponsor's current privacy level.
`previousSponsorsTier` (`SponsorsTier`)	The tier that the sponsorship used to use, for tier change events.
`sponsor` (`Sponsor`)	The user or organization who triggered this activity and was/is sponsoring the sponsorable.
`sponsorable` (`Sponsorable!`)	The user or organization that is being sponsored, the maintainer.
`sponsorsTier` (`SponsorsTier`)	The associated sponsorship tier.
`timestamp` (`DateTime`)	The timestamp of this event.
`viaBulkSponsorship` (`Boolean!`)	Was this sponsorship made alongside other sponsorships at the same time from the same sponsor?.

SponsorsActivityConnection

The connection type for SponsorsActivity.

Fields for `SponsorsActivityConnection`

Name	Description
`edges` (`[SponsorsActivityEdge]`)	A list of edges.

`nodes` (`[SponsorsActivity]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

SponsorsActivityEdge

An edge in a connection.

Fields for `SponsorsActivityEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`SponsorsActivity`)	The item at the end of the edge.

SponsorsGoal

A goal associated with a GitHub Sponsors listing, representing a target the sponsored maintainer would like to attain.

Fields for `SponsorsGoal`

Name	Description
`description` (`String`)	A description of the goal from the maintainer.
`kind` (`SponsorsGoalKind!`)	What the objective of this goal is.
`percentComplete` (`Int!`)	The percentage representing how complete this goal is, between 0-100.
`targetValue` (`Int!`)	What the goal amount is. Represents an amount in USD for monthly sponsorship amount goals. Represents a count of unique sponsors for total sponsors count goals.
`title` (`String!`)	A brief summary of the kind and target value of this goal.

SponsorsListing

A GitHub Sponsors listing.

`SponsorsListing` Implements

- `Node`

Fields for `SponsorsListing`

Name	Description
`activeGoal` (`SponsorsGoal`)	The current goal the maintainer is trying to reach with GitHub Sponsors, if any.
`activeStripeConnectAccount` (`StripeConnectAccount`)	The Stripe Connect account currently in use for payouts for this Sponsors listing, if any. Will only return a value when queried by the maintainer themselves, or by an admin of the sponsorable organization.
`billingCountryOrRegion` (`String`)	The name of the country or region with the maintainer's bank account or fiscal host. Will only return a value when queried by the maintainer themselves, or by an admin of the sponsorable organization.
`contactEmailAddress` (`String`)	The email address used by GitHub to contact the sponsorable about their GitHub Sponsors profile. Will only return a value when queried by the maintainer themselves, or by an admin of the sponsorable organization.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`dashboardResourcePath` (`URI!`)	The HTTP path for the Sponsors dashboard for this Sponsors listing.
`dashboardUrl` (`URI!`)	The HTTP URL for the Sponsors dashboard for this Sponsors listing.
`featuredItems` (`[SponsorsListingFeaturedItem!]!`)	The records featured on the GitHub Sponsors profile. **Arguments for `featuredItems`** `featureableTypes` (`[SponsorsListingFeaturedItemFeatureableType!]`) The types of featured items to return.
`fiscalHost` (`Organization`)	The fiscal host used for payments, if any. Will only return a value when queried by the maintainer themselves, or by an admin of the sponsorable organization.
`fullDescription` (`String!`)	The full description of the listing.
`fullDescriptionHTML` (`HTML!`)	The full description of the listing rendered to HTML.
`isPublic` (`Boolean!`)	Whether this listing is publicly visible.
`name` (`String!`)	The listing's full name.
`nextPayoutDate` (`Date`)	A future date on which this listing is eligible to receive a payout.
`residenceCountryOrRegion` (`String`)	The name of the country or region where the maintainer resides. Will only return a value when queried by the maintainer themselves, or by an admin of the sponsorable organization.

	by an admin of the sponsorable organization.
`resourcePath` (`URI!`)	The HTTP path for this Sponsors listing.
`shortDescription` (`String!`)	The short description of the listing.
`slug` (`String!`)	The short name of the listing.
`sponsorable` (`Sponsorable!`)	The entity this listing represents who can be sponsored on GitHub Sponsors.
`tiers` (`SponsorsTierConnection`)	The tiers for this GitHub Sponsors profile. **Arguments for `tiers`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `includeUnpublished` (`Boolean`) Whether to include tiers that aren't published. Only admins of the Sponsors listing can see draft tiers. Only admins of the Sponsors listing and viewers who are currently sponsoring on a retired tier can see those retired tiers. Defaults to including only published tiers, which are visible to anyone who can see the GitHub Sponsors profile. The default value is `false`. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`SponsorsTierOrder`) Ordering options for Sponsors tiers returned from the connection.
`url` (`URI!`)	The HTTP URL for this Sponsors listing.

SponsorsListingFeaturedItem

A record that is promoted on a GitHub Sponsors profile.

`SponsorsListingFeaturedItem` Implements

- `Node`

Fields for `SponsorsListingFeaturedItem`

Name	Description

`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`description` (`String`)	Will either be a description from the sponsorable maintainer about why they featured this item, or the item's description itself, such as a user's bio from their GitHub profile page.
`featureable` (`SponsorsListingFeatureableItem!`)	The record that is featured on the GitHub Sponsors profile.
`position` (`Int!`)	The position of this featured item on the GitHub Sponsors profile with a lower position indicating higher precedence. Starts at 1.
`sponsorsListing` (`SponsorsListing!`)	The GitHub Sponsors profile that features this record.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

SponsorsTier

A GitHub Sponsors tier associated with a GitHub Sponsors listing.

`SponsorsTier` Implements

- `Node`

Fields for `SponsorsTier`

Name	Description
`adminInfo` (`SponsorsTierAdminInfo`)	SponsorsTier information only visible to users that can administer the associated Sponsors listing.
`closestLesserValueTier` (`SponsorsTier`)	Get a different tier for this tier's maintainer that is at the same frequency as this tier but with an equal or lesser cost. Returns the published tier with the monthly price closest to this tier's without going over.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`description` (`String!`)	The description of the tier.
`descriptionHTML` (`HTML!`)	The tier description rendered to HTML.
`isCustomAmount` (`Boolean!`)	Whether this tier was chosen at checkout time by the sponsor rather than defined ahead of time by the maintainer who manages the Sponsors listing.
`isOneTime` (`Boolean!`)	Whether this tier is only for use with one-time sponsorships.
`monthlyPriceInCents` (`Int!`)	How much this tier costs per month in cents.

`monthlyPriceInDollars` (`Int!`)	How much this tier costs per month in USD.
`name` (`String!`)	The name of the tier.
`sponsorsListing` (`SponsorsListing!`)	The sponsors listing that this tier belongs to.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

SponsorsTierAdminInfo

SponsorsTier information only visible to users that can administer the associated Sponsors listing.

Fields for `SponsorsTierAdminInfo`

Name	Description
`isDraft` (`Boolean!`)	Indicates whether this tier is still a work in progress by the sponsorable and not yet published to the associated GitHub Sponsors profile. Draft tiers cannot be used for new sponsorships and will not be in use on existing sponsorships. Draft tiers cannot be seen by anyone but the admins of the GitHub Sponsors profile.
`isPublished` (`Boolean!`)	Indicates whether this tier is published to the associated GitHub Sponsors profile. Published tiers are visible to anyone who can see the GitHub Sponsors profile, and are available for use in sponsorships if the GitHub Sponsors profile is publicly visible.
`isRetired` (`Boolean!`)	Indicates whether this tier has been retired from the associated GitHub Sponsors profile. Retired tiers are no longer shown on the GitHub Sponsors profile and cannot be chosen for new sponsorships. Existing sponsorships may still use retired tiers if the sponsor selected the tier before it was retired.
`sponsorships` (`SponsorshipConnection!`)	The sponsorships using this tier. **Arguments for `sponsorships`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `includePrivate` (`Boolean`) Whether or not to return private sponsorships using this tier. Defaults to only returning public sponsorships on this

tier.
The default value is `false`.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`SponsorshipOrder`)
Ordering options for sponsorships returned from this connection. If left blank, the sponsorships will be ordered based on relevancy to the viewer.

SponsorsTierConnection

The connection type for SponsorsTier.

Fields for `SponsorsTierConnection`

Name	Description
`edges` (`[SponsorsTierEdge]`)	A list of edges.
`nodes` (`[SponsorsTier]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

SponsorsTierEdge

An edge in a connection.

Fields for `SponsorsTierEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`SponsorsTier`)	The item at the end of the edge.

Sponsorship

A sponsorship relationship between a sponsor and a maintainer.

`Sponsorship` Implements

- `Node`

Fields for `Sponsorship`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`isActive` (`Boolean!`)	Whether the sponsorship is active. False implies the sponsor is a past sponsor of the maintainer, while true implies they are a current sponsor.
`isOneTimePayment` (`Boolean!`)	Whether this sponsorship represents a one-time payment versus a recurring sponsorship.
`isSponsorOptedIntoEmail` (`Boolean`)	Whether the sponsor has chosen to receive sponsorship update emails sent from the sponsorable. Only returns a non-null value when the viewer has permission to know this.
`maintainer` (`User!`)	The entity that is being sponsored. **Deprecation notice** `maintainer` is deprecated. `Sponsorship.maintainer` will be removed. Use `Sponsorship.sponsorable` instead. Removal on 2020-04-01 UTC.
`privacyLevel` (`SponsorshipPrivacy!`)	The privacy level for this sponsorship.
`sponsor` (`User`)	The user that is sponsoring. Returns null if the sponsorship is private or if sponsor is not a user. **Deprecation notice** `sponsor` is deprecated. `Sponsorship.sponsor` will be removed. Use `Sponsorship.sponsorEntity` instead. Removal on 2020-10-01 UTC.
`sponsorEntity` (`Sponsor`)	The user or organization that is sponsoring, if you have permission to view them.
`sponsorable` (`Sponsorable!`)	The entity that is being sponsored.
`tier` (`SponsorsTier`)	The associated sponsorship tier.
`tierSelectedAt` (`DateTime`)	Identifies the date and time when the current tier was chosen for this sponsorship.

SponsorshipConnection

The connection type for Sponsorship.

Fields for `SponsorshipConnection`

Name	Description
`edges` (`[SponsorshipEdge]`)	A list of edges.

`edges` (`[SponsorshipEdge]`)	A list of edges.
`nodes` (`[Sponsorship]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.
`totalRecurringMonthlyPriceInCents` (`Int!`)	The total amount in cents of all recurring sponsorships in the connection whose amount you can view. Does not include one-time sponsorships.
`totalRecurringMonthlyPriceInDollars` (`Int!`)	The total amount in USD of all recurring sponsorships in the connection whose amount you can view. Does not include one-time sponsorships.

SponsorshipEdge

An edge in a connection.

Fields for `SponsorshipEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Sponsorship`)	The item at the end of the edge.

SponsorshipNewsletter

An update sent to sponsors of a user or organization on GitHub Sponsors.

`SponsorshipNewsletter` Implements

- `Node`

Fields for `SponsorshipNewsletter`

Name	Description
`author` (`User`)	The author of the newsletter.
`body` (`String!`)	The contents of the newsletter, the message the sponsorable wanted to give.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`isPublished` (`Boolean!`)	Indicates if the newsletter has been made available to sponsors.
`sponsorable` (`Sponsorable!`)	The user or organization this newsletter is from.
`subject` (`String!`)	The subject of the newsletter, what it's about.

`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

SponsorshipNewsletterConnection

The connection type for SponsorshipNewsletter.

Fields for `SponsorshipNewsletterConnection`

Name	Description
`edges` (`[SponsorshipNewsletterEdge]`)	A list of edges.
`nodes` (`[SponsorshipNewsletter]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

SponsorshipNewsletterEdge

An edge in a connection.

Fields for `SponsorshipNewsletterEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`SponsorshipNewsletter`)	The item at the end of the edge.

SshSignature

Represents an SSH signature on a Commit or Tag.

`SshSignature` Implements

- `GitSignature`

Fields for `SshSignature`

Name	Description
`email` (`String!`)	Email used to sign this object.
`isValid` (`Boolean!`)	True if the signature is valid and verified by GitHub.
`keyFingerprint` (`String`)	Hex-encoded fingerprint of the key that signed this object.
`payload` (`String!`)	Payload for GPG signing object. Raw ODB object

`payload` (`String!`)	Payload verified signature signature payload. This is the commit object without the signature header.
`signature` (`String!`)	ASCII-armored signature header from object.
`signer` (`User`)	GitHub user corresponding to the email signing this commit.
`state` (`GitSignatureState!`)	The state of this signature. `VALID` if signature is valid and verified by GitHub, otherwise represents reason why signature is considered invalid.
`wasSignedByGitHub` (`Boolean!`)	True if the signature was made with GitHub's signing key.

StargazerConnection

The connection type for User.

Fields for `StargazerConnection`

Name	Description
`edges` (`[StargazerEdge]`)	A list of edges.
`nodes` (`[User]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

StargazerEdge

Represents a user that's starred a repository.

Fields for `StargazerEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`starredAt` (`DateTime!`)	Identifies when the item was starred.

StarredRepositoryConnection

The connection type for Repository.

Fields for `StarredRepositoryConnection`

Name	Description
`edges` (`[StarredRepositoryEdge]`)	A list of edges.
`isOverLimit` (`Boolean!`)	Is the list of stars for this user truncated? This is

`isOverLimit` (`Boolean!`)	Is the list of stars for this user truncated? This is true for users that have many stars.
`nodes` (`[Repository]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

StarredRepositoryEdge

Represents a starred repository.

Fields for `StarredRepositoryEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`starredAt` (`DateTime!`)	Identifies when the item was starred.

Status

Represents a commit status.

`Status` Implements

- `Node`

Fields for `Status`

Name	Description
`combinedContexts` (`StatusCheckRollupContextConnection!`)	A list of status contexts and check runs for this commit. **Arguments for `combinedContexts`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`commit` (`Commit`)	The commit this status is attached to.

`context` (`StatusContext`)	Looks up an individual status context by context name. **Arguments for `context`** `name` (`String!`) The context name.
`contexts` (`[StatusContext!]!`)	The individual status contexts for this commit.
`state` (`StatusState!`)	The combined commit status.

StatusCheckConfiguration

Required status check.

Fields for `StatusCheckConfiguration`

Name	Description
`context` (`String!`)	The status check context name that must be present on the commit.
`integrationId` (`Int`)	The optional integration ID that this status check must originate from.

StatusCheckRollup

Represents the rollup for both the check runs and status for a commit.

`StatusCheckRollup` Implements

- `Node`

Fields for `StatusCheckRollup`

Name	Description
`commit` (`Commit`)	The commit the status and check runs are attached to.
`contexts` (`StatusCheckRollupContextConnection!`)	A list of status contexts and check runs for this commit. **Arguments for `contexts`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`state` (`StatusState!`)	The combined status for the commit.

StatusCheckRollupContextConnection

The connection type for StatusCheckRollupContext.

Fields for `StatusCheckRollupContextConnection`

Name	Description
`checkRunCount` (`Int!`)	The number of check runs in this rollup.
`checkRunCountsByState` (`[CheckRunStateCount!]`)	Counts of check runs by state.
`edges` (`[StatusCheckRollupContextEdge]`)	A list of edges.
`nodes` (`[StatusCheckRollupContext]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`statusContextCount` (`Int!`)	The number of status contexts in this rollup.
`statusContextCountsByState` (`[StatusContextStateCount!]`)	Counts of status contexts by state.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

StatusCheckRollupContextEdge

An edge in a connection.

Fields for `StatusCheckRollupContextEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`StatusCheckRollupContext`)	The item at the end of the edge.

StatusContext

Represents an individual commit status context.

`StatusContext` Implements

- `Node`
- `RequirableByPullRequest`

Fields for `StatusContext`

Name	Description
`avatarUrl` (`URI`)	The avatar of the OAuth application or the user that created the status. **Arguments for `avatarUrl`** `size` (`Int`) The size of the resulting square image. The default value is `40`.
`commit` (`Commit`)	This commit this status context is attached to.
`context` (`String!`)	The name of this status context.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`creator` (`Actor`)	The actor who created this status context.
`description` (`String`)	The description for this status context.
`isRequired` (`Boolean!`)	Whether this is required to pass before merging for a specific pull request. **Arguments for `isRequired`** `pullRequestId` (`ID`) The id of the pull request this is required for. `pullRequestNumber` (`Int`) The number of the pull request this is required for.
`state` (`StatusState!`)	The state of this status context.
`targetUrl` (`URI`)	The URL for this status context.

StatusContextStateCount

Represents a count of the state of a status context.

Fields for `StatusContextStateCount`

Name	Description

`count` (`Int!`)	The number of statuses with this state.
`state` (`StatusState!`)	The state of a status context.

StripeConnectAccount

A Stripe Connect account for receiving sponsorship funds from GitHub Sponsors.

Fields for `StripeConnectAccount`

Name	Description
`accountId` (`String!`)	The account number used to identify this Stripe Connect account.
`billingCountryOrRegion` (`String`)	The name of the country or region of an external account, such as a bank account, tied to the Stripe Connect account. Will only return a value when queried by the maintainer of the associated GitHub Sponsors profile themselves, or by an admin of the sponsorable organization.
`countryOrRegion` (`String`)	The name of the country or region of the Stripe Connect account. Will only return a value when queried by the maintainer of the associated GitHub Sponsors profile themselves, or by an admin of the sponsorable organization.
`isActive` (`Boolean!`)	Whether this Stripe Connect account is currently in use for the associated GitHub Sponsors profile.
`sponsorsListing` (`SponsorsListing!`)	The GitHub Sponsors profile associated with this Stripe Connect account.
`stripeDashboardUrl` (`URI!`)	The URL to access this Stripe Connect account on Stripe's website.

Submodule

A pointer to a repository at a specific revision embedded inside another repository.

Fields for `Submodule`

Name	Description
`branch` (`String`)	The branch of the upstream submodule for tracking updates.
`gitUrl` (`URI!`)	The git URL of the submodule repository.
`name` (`String!`)	The name of the submodule in .gitmodules.
`nameRaw` (`Base64String!`)	The name of the submodule in .gitmodules (Base64-encoded).
`path` (`String!`)	The path in the superproject that this submodule is located in.

`pathRaw` (`Base64String!`)	The path in the superproject that this submodule is located in (Base64-encoded).
`subprojectCommitOid` (`GitObjectID`)	The commit revision of the subproject repository being tracked by the submodule.

SubmoduleConnection

The connection type for Submodule.

Fields for `SubmoduleConnection`

Name	Description
`edges` (`[SubmoduleEdge]`)	A list of edges.
`nodes` (`[Submodule]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

SubmoduleEdge

An edge in a connection.

Fields for `SubmoduleEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Submodule`)	The item at the end of the edge.

SubscribedEvent

Represents a `subscribed` event on a given `Subscribable`.

`SubscribedEvent` Implements

- `Node`

Fields for `SubscribedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`subscribable` (`Subscribable!`)	Object referenced by event.

SuggestedReviewer

A suggestion to review a pull request based on a user's commit history and review comments.

Fields for `SuggestedReviewer`

Name	Description
`isAuthor` (`Boolean!`)	Is this suggestion based on past commits?.
`isCommenter` (`Boolean!`)	Is this suggestion based on past review comments?.
`reviewer` (`User!`)	Identifies the user suggested to review the pull request.

Tag

Represents a Git tag.

`Tag` Implements

- `GitObject`
- `Node`

Fields for `Tag`

Name	Description
`abbreviatedOid` (`String!`)	An abbreviated version of the Git object ID.
`commitResourcePath` (`URI!`)	The HTTP path for this Git object.
`commitUrl` (`URI!`)	The HTTP URL for this Git object.
`message` (`String`)	The Git tag message.
`name` (`String!`)	The Git tag name.
`oid` (`GitObjectID!`)	The Git object ID.
`repository` (`Repository!`)	The Repository the Git object belongs to.
`tagger` (`GitActor`)	Details about the tag author.
`target` (`GitObject!`)	The Git object the tag points to.

TagNamePatternParameters

Parameters to be used for the tag_name_pattern rule.

Fields for `TagNamePatternParameters`

Name	Description
`name` (`String`)	How this rule will appear to users.
`negate` (`Boolean!`)	If true, the rule will fail if the pattern matches.
`operator` (`String!`)	The operator to use for matching.
`pattern` (`String!`)	The pattern to match with.

Team

A team of users in an organization.

`Team` Implements

- `MemberStatusable`
- `Node`
- `Subscribable`

Fields for `Team`

Name	Description
`ancestors` (`TeamConnection!`)	A list of teams that are ancestors of this team. **Arguments for `ancestors`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`avatarUrl` (`URI`)	A URL pointing to the team's avatar. **Arguments for `avatarUrl`** `size` (`Int`) The size in pixels of the resulting square image. The default value is `400` .

`childTeams` (`TeamConnection!`)	List of child teams belonging to this team. **Arguments for `childTeams`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `immediateOnly` (`Boolean`) Whether to list immediate child teams or all descendant child teams. The default value is `true` . `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`TeamOrder`) Order for connection. `userLogins` (`[String!]`) User logins to filter by.
`combinedSlug` (`String!`)	The slug corresponding to the organization and team.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`description` (`String`)	The description of the team.
`discussion` (`TeamDiscussion`)	Find a team discussion by its number. **Arguments for `discussion`** `number` (`Int!`) The sequence number of the discussion to find.
`discussions` (`TeamDiscussionConnection!`)	A list of team discussions. **Arguments for `discussions`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`isPinned` (`Boolean`)
If provided, filters discussions according to whether or not they are pinned.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`TeamDiscussionOrder`)
Order for connection.

`discussionsResourcePath` (`URI!`)	The HTTP path for team discussions.
`discussionsUrl` (`URI!`)	The HTTP URL for team discussions.
`editTeamResourcePath` (`URI!`)	The HTTP path for editing this team.
`editTeamUrl` (`URI!`)	The HTTP URL for editing this team.
`invitations` (`OrganizationInvitationConnection`)	A list of pending invitations for users to this team.

Arguments for `invitations`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`memberStatuses` (`UserStatusConnection!`)	Get the status messages members of this entity have set that are either public or visible only to the organization.

Arguments for `memberStatuses`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`UserStatusOrder`)
Ordering options for user statuses returned from the connection.

`members` (`TeamMemberConnection!`)	A list of users who are members of this team. **Arguments for `members`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `membership` (`TeamMembershipType`) Filter by membership type. The default value is `ALL`. `orderBy` (`TeamMemberOrder`) Order for the connection. `query` (`String`) The search string to look for. `role` (`TeamMemberRole`) Filter by team member role.
`membersResourcePath` (`URI!`)	The HTTP path for the team' members.
`membersUrl` (`URI!`)	The HTTP URL for the team' members.
`name` (`String!`)	The name of the team.
`newTeamResourcePath` (`URI!`)	The HTTP path creating a new team.
`newTeamUrl` (`URI!`)	The HTTP URL creating a new team.
`notificationSetting` (`TeamNotificationSetting!`)	The notification setting that the team has set.
`organization` (`Organization!`)	The organization that owns this team.
`parentTeam` (`Team`)	The parent team of the team.
`privacy` (`TeamPrivacy!`)	The level of privacy the team has.
`projectV2` (`ProjectV2`)	Finds and returns the project according to the provided project number. **Arguments for `projectV2`**

`number` (`Int!`)
The Project number.

`projectsV2` (`ProjectV2Connection!`)

List of projects this team has collaborator access to.

Arguments for `projectsV2`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`filterBy` (`ProjectV2Filters`)
Filtering options for projects returned from this connection.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ProjectV2Order`)
How to order the returned projects.

`query` (`String`)
The query to search projects by.
The default value is `` .

`repositories` (`TeamRepositoryConnection!`)

A list of repositories this team has access to.

Arguments for `repositories`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`TeamRepositoryOrder`)
Order for the connection.

`query` (`String`)
The search string to look for. Repositories will be returned where the name contains your search string.

`repositoriesResourcePath` (`URI!`)	The HTTP path for this team's repositories.
`repositoriesUrl` (`URI!`)	The HTTP URL for this team's repositories.
`resourcePath` (`URI!`)	The HTTP path for this team.
`reviewRequestDelegationAlgorithm` (`TeamReviewAssignmentAlgorithm`)	What algorithm is used for review assignment for this team. **Preview notice** `reviewRequestDelegationAlgorithm` is available under the Team review assignments preview. During the preview period, the API may change without notice.
`reviewRequestDelegationEnabled` (`Boolean!`)	True if review assignment is enabled for this team. **Preview notice** `reviewRequestDelegationEnabled` is available under the Team review assignments preview. During the preview period, the API may change without notice.
`reviewRequestDelegationMemberCount` (`Int`)	How many team members are required for review assignment for this team. **Preview notice** `reviewRequestDelegationMemberCount` is available under the Team review assignments preview. During the preview period, the API may change without notice.
`reviewRequestDelegationNotifyTeam` (`Boolean!`)	When assigning team members via delegation, whether the entire team should be notified as well. **Preview notice** `reviewRequestDelegationNotifyTeam` is available under the Team review assignments preview. During the preview period, the API may change without notice.
`slug` (`String!`)	The slug corresponding to the team.
`teamsResourcePath` (`URI!`)	The HTTP path for this team's teams.
`teamsUrl` (`URI!`)	The HTTP URL for this team's teams.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this team.
`viewerCanAdminister` (`Boolean!`)	Team is adminable by the viewer.

`viewerCanAdminister` (`Boolean!`)	Team is adminable by the viewer.
`viewerCanSubscribe` (`Boolean!`)	Check if the viewer is able to change their subscription status for the repository.
`viewerSubscription` (`SubscriptionState`)	Identifies if the viewer is watching, not watching, or ignoring the subscribable entity.

TeamAddMemberAuditEntry

Audit log entry for a team.add_member event.

`TeamAddMemberAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `TeamAuditEntryData`

Fields for `TeamAddMemberAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`isLdapMapped` (`Boolean`)	Whether the team was mapped to an LDAP Group.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`team` (`Team`)	The team associated with the action.
`teamName` (`String`)	The name of the team.

`teamResourcePath` (`URI`)	The HTTP path for this team.
`teamUrl` (`URI`)	The HTTP URL for this team.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

TeamAddRepositoryAuditEntry

Audit log entry for a team.add_repository event.

`TeamAddRepositoryAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`
- `TeamAuditEntryData`

Fields for `TeamAddRepositoryAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`isLdapMapped` (`Boolean`)	Whether the team was mapped to an LDAP Group.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.

`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`team` (`Team`)	The team associated with the action.
`teamName` (`String`)	The name of the team.
`teamResourcePath` (`URI`)	The HTTP path for this team.
`teamUrl` (`URI`)	The HTTP URL for this team.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

TeamChangeParentTeamAuditEntry

Audit log entry for a team.change_parent_team event.

`TeamChangeParentTeamAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `TeamAuditEntryData`

Fields for `TeamChangeParentTeamAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.

`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`isLdapMapped` (`Boolean`)	Whether the team was mapped to an LDAP Group.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`parentTeam` (`Team`)	The new parent team.
`parentTeamName` (`String`)	The name of the new parent team.
`parentTeamNameWas` (`String`)	The name of the former parent team.
`parentTeamResourcePath` (`URI`)	The HTTP path for the parent team.
`parentTeamUrl` (`URI`)	The HTTP URL for the parent team.
`parentTeamWas` (`Team`)	The former parent team.
`parentTeamWasResourcePath` (`URI`)	The HTTP path for the previous parent team.
`parentTeamWasUrl` (`URI`)	The HTTP URL for the previous parent team.
`team` (`Team`)	The team associated with the action.
`teamName` (`String`)	The name of the team.
`teamResourcePath` (`URI`)	The HTTP path for this team.
`teamUrl` (`URI`)	The HTTP URL for this team.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

TeamConnection

The connection type for Team.

Fields for `TeamConnection`

Name	Description
`edges` (`[TeamEdge]`)	A list of edges.

`nodes` (`[Team]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

TeamDiscussion

A team discussion.

`TeamDiscussion` Implements

- `Comment`
- `Deletable`
- `Node`
- `Reactable`
- `Subscribable`
- `UniformResourceLocatable`
- `Updatable`
- `UpdatableComment`

Fields for `TeamDiscussion`

Name	Description
`author` (`Actor`)	The actor who authored the comment.
`authorAssociation` (`CommentAuthorAssociation!`)	Author's association with the discussion's team. **Deprecation notice** `authorAssociation` is deprecated. The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.
`body` (`String!`)	The body as Markdown.
`bodyHTML` (`HTML!`)	The body rendered to HTML.
`bodyText` (`String!`)	The body rendered to text.
`bodyVersion` (`String!`)	Identifies the discussion body hash. **Deprecation notice** `bodyVersion` is deprecated. The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.

`comments` (`TeamDiscussionCommentConnection!`)

A list of comments on this discussion.

Deprecation notice
`comments` is deprecated.
The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.

Arguments for `comments`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`fromComment` (`Int`)
When provided, filters the connection such that results begin with the comment with this number.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`TeamDiscussionCommentOrder`)
Order for connection.

`commentsResourcePath` (`URI!`)

The HTTP path for discussion comments.

Deprecation notice
`commentsResourcePath` is deprecated.
The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.

`commentsUrl` (`URI!`)

The HTTP URL for discussion comments.

Deprecation notice
`commentsUrl` is deprecated.
The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a

	suitable replacement. Removal on 2024-07-01 UTC.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`createdViaEmail` (`Boolean!`)	Check if this comment was created via an email reply.
`databaseId` (`Int`)	Identifies the primary key from the database.
`editor` (`Actor`)	The actor who edited the comment.
`includesCreatedEdit` (`Boolean!`)	Check if this comment was edited and includes an edit with the creation data.
`isPinned` (`Boolean!`)	Whether or not the discussion is pinned. **Deprecation notice** `isPinned` is deprecated. The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.
`isPrivate` (`Boolean!`)	Whether or not the discussion is only visible to team members and org admins. **Deprecation notice** `isPrivate` is deprecated. The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.
`lastEditedAt` (`DateTime`)	The moment the editor made the last edit.
`number` (`Int!`)	Identifies the discussion within its team. **Deprecation notice** `number` is deprecated. The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.

`publishedAt` (`DateTime`)	Identifies when the comment was published at.
`reactionGroups` (`[ReactionGroup!]`)	A list of reactions grouped by content left on the subject.
`reactions` (`ReactionConnection!`)	A list of Reactions left on the Issue. **Arguments for `reactions`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `content` (`ReactionContent`) Allows filtering Reactions by emoji. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`ReactionOrder`) Allows specifying the order in which reactions are returned.
`resourcePath` (`URI!`)	The HTTP path for this discussion. **Deprecation notice** `resourcePath` is deprecated. The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.
`team` (`Team!`)	The team that defines the context of this discussion. **Deprecation notice** `team` is deprecated. The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.
`title` (`String!`)	The title of the discussion. **Deprecation notice**

Deprecation notice
`title` is deprecated.
The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.

`updatedAt` (`DateTime!`)

Identifies the date and time when the object was last updated.

`url` (`URI!`)

The HTTP URL for this discussion.

Deprecation notice
`url` is deprecated.
The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.

`userContentEdits` (`UserContentEditConnection`)

A list of edits to this content.

Arguments for `userContentEdits`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`viewerCanDelete` (`Boolean!`)

Check if the current viewer can delete this object.

`viewerCanPin` (`Boolean!`)

Whether or not the current viewer can pin this discussion.

Deprecation notice
`viewerCanPin` is deprecated.
The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.

`viewerCanReact` (`Boolean!`)	Can user react to this subject.
`viewerCanSubscribe` (`Boolean!`)	Check if the viewer is able to change their subscription status for the repository.
`viewerCanUpdate` (`Boolean!`)	Check if the current viewer can update this object.
`viewerCannotUpdateReasons` (`[CommentCannotUpdateReason!]!`)	Reasons why the current viewer can not update this comment.
`viewerDidAuthor` (`Boolean!`)	Did the viewer author this comment.
`viewerSubscription` (`SubscriptionState`)	Identifies if the viewer is watching, not watching, or ignoring the subscribable entity.

TeamDiscussionComment

A comment on a team discussion.

`TeamDiscussionComment` Implements

- `Comment`
- `Deletable`
- `Node`
- `Reactable`
- `UniformResourceLocatable`
- `Updatable`
- `UpdatableComment`

Fields for `TeamDiscussionComment`

Name	Description
`author` (`Actor`)	The actor who authored the comment.
`authorAssociation` (`CommentAuthorAssociation!`)	Author's association with the comment's team. **Deprecation notice** `authorAssociation` is deprecated. The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.
`body` (`String!`)	The body as Markdown.
`bodyHTML` (`HTML!`)	The body rendered to HTML.
`bodyText` (`String!`)	The body rendered to text.
`bodyVersion` (`String!`)	The current version of the body content.

Deprecation notice
`bodyVersion` is deprecated.
The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.

`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`createdViaEmail` (`Boolean!`)	Check if this comment was created via an email reply.
`databaseId` (`Int`)	Identifies the primary key from the database.
`discussion` (`TeamDiscussion!`)	The discussion this comment is about.

Deprecation notice
`discussion` is deprecated.
The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.

`editor` (`Actor`)	The actor who edited the comment.
`includesCreatedEdit` (`Boolean!`)	Check if this comment was edited and includes an edit with the creation data.
`lastEditedAt` (`DateTime`)	The moment the editor made the last edit.
`number` (`Int!`)	Identifies the comment number.

Deprecation notice
`number` is deprecated.
The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.

`publishedAt` (`DateTime`)	Identifies when the comment was published at.
`reactionGroups` (`[ReactionGroup!]`)	A list of reactions grouped by content left on the subject.
`reactions` (`ReactionConnection!`)	A list of Reactions left on the Issue.

Arguments for `reactions`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`content` (`ReactionContent`)
Allows filtering Reactions by emoji.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ReactionOrder`)
Allows specifying the order in which reactions are returned.

`resourcePath` (`URI!`)

The HTTP path for this comment.

Deprecation notice
`resourcePath` is deprecated.
The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.

`updatedAt` (`DateTime!`)

Identifies the date and time when the object was last updated.

`url` (`URI!`)

The HTTP URL for this comment.

Deprecation notice
`url` is deprecated.
The Team Discussions feature is deprecated in favor of Organization Discussions. Follow the guide at https://github.blog/changelog/2023-02-08-sunset-notice-team-discussions/ to find a suitable replacement. Removal on 2024-07-01 UTC.

`userContentEdits` (`UserContentEditConnection`)

A list of edits to this content.

Arguments for `userContentEdits`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

	`first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`viewerCanDelete` (`Boolean!`)	Check if the current viewer can delete this object.
`viewerCanReact` (`Boolean!`)	Can user react to this subject.
`viewerCanUpdate` (`Boolean!`)	Check if the current viewer can update this object.
`viewerCannotUpdateReasons` (`[CommentCannotUpdateReason!]!`)	Reasons why the current viewer can not update this comment.
`viewerDidAuthor` (`Boolean!`)	Did the viewer author this comment.

TeamDiscussionCommentConnection

The connection type for TeamDiscussionComment.

Fields for `TeamDiscussionCommentConnection`

Name	Description
`edges` (`[TeamDiscussionCommentEdge]`)	A list of edges.
`nodes` (`[TeamDiscussionComment]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

TeamDiscussionCommentEdge

An edge in a connection.

Fields for `TeamDiscussionCommentEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`TeamDiscussionComment`)	The item at the end of the edge.

TeamDiscussionConnection

The connection type for TeamDiscussion.

Fields for `TeamDiscussionConnection`

Name	Description
`edges` (`[TeamDiscussionEdge]`)	A list of edges.
`nodes` (`[TeamDiscussion]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

TeamDiscussionEdge

An edge in a connection.

Fields for `TeamDiscussionEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`TeamDiscussion`)	The item at the end of the edge.

TeamEdge

An edge in a connection.

Fields for `TeamEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`Team`)	The item at the end of the edge.

TeamMemberConnection

The connection type for User.

Fields for `TeamMemberConnection`

Name	Description
`edges` (`[TeamMemberEdge]`)	A list of edges.
`nodes` (`[User]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

TeamMemberEdge

Represents a user who is a member of a team.

Fields for `TeamMemberEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`memberAccessResourcePath` (`URI!`)	The HTTP path to the organization's member access page.
`memberAccessUrl` (`URI!`)	The HTTP URL to the organization's member access page.
`role` (`TeamMemberRole!`)	The role the member has on the team.

TeamRemoveMemberAuditEntry

Audit log entry for a team.remove_member event.

`TeamRemoveMemberAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `TeamAuditEntryData`

Fields for `TeamRemoveMemberAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.
`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`isLdapMapped` (`Boolean`)	Whether the team was mapped to an LDAP Group.
`operationType` (`OperationType`)	The corresponding operation type for the action.

`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`team` (`Team`)	The team associated with the action.
`teamName` (`String`)	The name of the team.
`teamResourcePath` (`URI`)	The HTTP path for this team.
`teamUrl` (`URI`)	The HTTP URL for this team.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

TeamRemoveRepositoryAuditEntry

Audit log entry for a team.remove_repository event.

`TeamRemoveRepositoryAuditEntry` Implements

- `AuditEntry`
- `Node`
- `OrganizationAuditEntryData`
- `RepositoryAuditEntryData`
- `TeamAuditEntryData`

Fields for `TeamRemoveRepositoryAuditEntry`

Name	Description
`action` (`String!`)	The action name.
`actor` (`AuditEntryActor`)	The user who initiated the action.
`actorIp` (`String`)	The IP address of the actor.
`actorLocation` (`ActorLocation`)	A readable representation of the actor's location.
`actorLogin` (`String`)	The username of the user who initiated the action.
`actorResourcePath` (`URI`)	The HTTP path for the actor.
`actorUrl` (`URI`)	The HTTP URL for the actor.

`createdAt` (`PreciseDateTime!`)	The time the action was initiated.
`isLdapMapped` (`Boolean`)	Whether the team was mapped to an LDAP Group.
`operationType` (`OperationType`)	The corresponding operation type for the action.
`organization` (`Organization`)	The Organization associated with the Audit Entry.
`organizationName` (`String`)	The name of the Organization.
`organizationResourcePath` (`URI`)	The HTTP path for the organization.
`organizationUrl` (`URI`)	The HTTP URL for the organization.
`repository` (`Repository`)	The repository associated with the action.
`repositoryName` (`String`)	The name of the repository.
`repositoryResourcePath` (`URI`)	The HTTP path for the repository.
`repositoryUrl` (`URI`)	The HTTP URL for the repository.
`team` (`Team`)	The team associated with the action.
`teamName` (`String`)	The name of the team.
`teamResourcePath` (`URI`)	The HTTP path for this team.
`teamUrl` (`URI`)	The HTTP URL for this team.
`user` (`User`)	The user affected by the action.
`userLogin` (`String`)	For actions involving two users, the actor is the initiator and the user is the affected user.
`userResourcePath` (`URI`)	The HTTP path for the user.
`userUrl` (`URI`)	The HTTP URL for the user.

TeamRepositoryConnection

The connection type for Repository.

Fields for `TeamRepositoryConnection`

Name	Description
`edges` (`[TeamRepositoryEdge]`)	A list of edges.
`nodes` (`[Repository]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

TeamRepositoryEdge

Represents a team repository.

Fields for `TeamRepositoryEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`permission` (`RepositoryPermission!`)	The permission level the team has on the repository.

TextMatch

A text match within a search result.

Fields for `TextMatch`

Name	Description
`fragment` (`String!`)	The specific text fragment within the property matched on.
`highlights` (`[TextMatchHighlight!]!`)	Highlights within the matched fragment.
`property` (`String!`)	The property matched on.

TextMatchHighlight

Represents a single highlight in a search result match.

Fields for `TextMatchHighlight`

Name	Description
`beginIndice` (`Int!`)	The indice in the fragment where the matched text begins.
`endIndice` (`Int!`)	The indice in the fragment where the matched text ends.
`text` (`String!`)	The text matched.

Topic

A topic aggregates entities that are related to a subject.

`Topic` Implements

- `Node`
- `Starrable`

Fields for `Topic`

Name	Description
`name` (`String!`)	The topic's name.
`relatedTopics` (`[Topic!]!`)	A list of related topics, including aliases of this topic, sorted with the most relevant first. Returns up to 10 Topics. **Arguments for `relatedTopics`** `first` (`Int`) How many topics to return. The default value is `3`.
`repositories` (`RepositoryConnection!`)	A list of repositories. **Arguments for `repositories`** `affiliations` (`[RepositoryAffiliation]`) Array of viewer's affiliation options for repositories returned from the connection. For example, OWNER will include only repositories that the current viewer owns. `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `hasIssuesEnabled` (`Boolean`) If non-null, filters repositories according to whether they have issues enabled. `isLocked` (`Boolean`) If non-null, filters repositories according to whether they have been locked. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`RepositoryOrder`) Ordering options for repositories returned from the connection. `ownerAffiliations` (`[RepositoryAffiliation]`) Array of owner's affiliation options for repositories returned from the connection. For example, OWNER will include only repositories that the organization or user being viewed owns. `privacy` (`RepositoryPrivacy`) If non-null, filters repositories according to

	If non-null, filters repositories according to privacy. `sponsorableOnly` (`Boolean`) If true, only repositories whose owner can be sponsored via GitHub Sponsors will be returned. The default value is `false`.
`stargazerCount` (`Int!`)	Returns a count of how many stargazers there are on this object.
`stargazers` (`StargazerConnection!`)	A list of users who have starred this starrable. **Arguments for `stargazers`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`StarOrder`) Order for connection.
`viewerHasStarred` (`Boolean!`)	Returns a boolean indicating whether the viewing user has starred this starrable.

TransferredEvent

Represents a `transferred` event on a given issue or pull request.

`TransferredEvent` Implements

- `Node`

Fields for `TransferredEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`fromRepository` (`Repository`)	The repository this came from.
`issue` (`Issue!`)	Identifies the issue associated with the event.

Tree

Represents a Git tree.

`Tree` Implements

- `GitObject`
- `Node`

Fields for `Tree`

Name	Description
`abbreviatedOid` (`String!`)	An abbreviated version of the Git object ID.
`commitResourcePath` (`URI!`)	The HTTP path for this Git object.
`commitUrl` (`URI!`)	The HTTP URL for this Git object.
`entries` (`[TreeEntry!]`)	A list of tree entries.
`oid` (`GitObjectID!`)	The Git object ID.
`repository` (`Repository!`)	The Repository the Git object belongs to.

TreeEntry

Represents a Git tree entry.

Fields for `TreeEntry`

Name	Description
`extension` (`String`)	The extension of the file.
`isGenerated` (`Boolean!`)	Whether or not this tree entry is generated.
`language` (`Language`)	The programming language this file is written in.
`lineCount` (`Int`)	Number of lines in the file.
`mode` (`Int!`)	Entry file mode.
`name` (`String!`)	Entry file name.
`nameRaw` (`Base64String!`)	Entry file name. (Base64-encoded).
`object` (`GitObject`)	Entry file object.
`oid` (`GitObjectID!`)	Entry file Git object ID.
`path` (`String`)	The full path of the file.
`pathRaw` (`Base64String`)	The full path of the file. (Base64-encoded).

`repository` (`Repository!`)	The Repository the tree entry belongs to.
`size` (`Int!`)	Entry byte size.
`submodule` (`Submodule`)	If the TreeEntry is for a directory occupied by a submodule project, this returns the corresponding submodule.
`type` (`String!`)	Entry file type.

UnassignedEvent

Represents an `unassigned` event on any assignable object.

`UnassignedEvent` Implements

- `Node`

Fields for `UnassignedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`assignable` (`Assignable!`)	Identifies the assignable associated with the event.
`assignee` (`Assignee`)	Identifies the user or mannequin that was unassigned.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`user` (`User`)	Identifies the subject (user) who was unassigned. **Deprecation notice** `user` is deprecated. Assignees can now be mannequins. Use the `assignee` field instead. Removal on 2020-01-01 UTC.

UnknownSignature

Represents an unknown signature on a Commit or Tag.

`UnknownSignature` Implements

- `GitSignature`

Fields for `UnknownSignature`

Name	Description

`email` (`String!`)	Email used to sign this object.
`isValid` (`Boolean!`)	True if the signature is valid and verified by GitHub.
`payload` (`String!`)	Payload for GPG signing object. Raw ODB object without the signature header.
`signature` (`String!`)	ASCII-armored signature header from object.
`signer` (`User`)	GitHub user corresponding to the email signing this commit.
`state` (`GitSignatureState!`)	The state of this signature. `VALID` if signature is valid and verified by GitHub, otherwise represents reason why signature is considered invalid.
`wasSignedByGitHub` (`Boolean!`)	True if the signature was made with GitHub's signing key.

UnlabeledEvent

Represents an `unlabeled` event on a given issue or pull request.

`UnlabeledEvent` Implements

- `Node`

Fields for `UnlabeledEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`label` (`Label!`)	Identifies the label associated with the `unlabeled` event.
`labelable` (`Labelable!`)	Identifies the `Labelable` associated with the event.

UnlockedEvent

Represents an `unlocked` event on a given issue or pull request.

`UnlockedEvent` Implements

- `Node`

Fields for `UnlockedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.

`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`lockable` (`Lockable!`)	Object that was unlocked.

UnmarkedAsDuplicateEvent

Represents an `unmarked_as_duplicate` event on a given issue or pull request.

`UnmarkedAsDuplicateEvent` Implements

- `Node`

Fields for `UnmarkedAsDuplicateEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`canonical` (`IssueOrPullRequest`)	The authoritative issue or pull request which has been duplicated by another.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`duplicate` (`IssueOrPullRequest`)	The issue or pull request which has been marked as a duplicate of another.
`isCrossRepository` (`Boolean!`)	Canonical and duplicate belong to different repositories.

UnpinnedEvent

Represents an `unpinned` event on a given issue or pull request.

`UnpinnedEvent` Implements

- `Node`

Fields for `UnpinnedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`issue` (`Issue!`)	Identifies the issue associated with the event.

UnsubscribedEvent

Represents an `unsubscribed` event on a given `Subscribable`.

`UnsubscribedEvent` Implements

- `Node`

Fields for `UnsubscribedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`subscribable` (`Subscribable!`)	Object referenced by event.

UpdateParameters

Only allow users with bypass permission to update matching refs.

Fields for `UpdateParameters`

Name	Description
`updateAllowsFetchAndMerge` (`Boolean!`)	Branch can pull changes from its upstream repository.

User

A user is an individual's account on GitHub that owns repositories and can make new content.

`User` Implements

- `Actor`
- `Node`
- `PackageOwner`
- `ProfileOwner`
- `ProjectOwner`
- `ProjectV2Owner`
- `ProjectV2Recent`
- `RepositoryDiscussionAuthor`
- `RepositoryDiscussionCommentAuthor`
- `RepositoryOwner`
- `Sponsorable`
- `UniformResourceLocatable`

Fields for `User`

Name	Description
[illegible] ([illegible])	Determine if this repository owner has any items [illegible]

`anyPinnableItems` (`Boolean!`)	Determine if this repository owner has any items that can be pinned to their profile. **Arguments for `anyPinnableItems`** `type` (`PinnableItemType`) Filter to only a particular kind of pinnable item.
`avatarUrl` (`URI!`)	A URL pointing to the user's public avatar. **Arguments for `avatarUrl`** `size` (`Int`) The size of the resulting square image.
`bio` (`String`)	The user's public profile bio.
`bioHTML` (`HTML!`)	The user's public profile bio as HTML.
`canReceiveOrganizationEmailsWhenNotificationsRestricted` (`Boolean!`)	Could this user receive email notifications, if the organization had notification restrictions enabled?. **Arguments for `canReceiveOrganizationEmailsWhenNotificationsRestricted`** `login` (`String!`) The login of the organization to check.
`commitComments` (`CommitCommentConnection!`)	A list of commit comments made by this user. **Arguments for `commitComments`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`company` (`String`)	The user's public profile company.
`companyHTML` (`HTML!`)	The user's public profile company as HTML.
`contributionsCollection` (`ContributionsCollection!`)	The collection of contributions this user has made to different repositories.

Arguments for `contributionsCollection`

`from` (`DateTime`)
Only contributions made at this time or later will be counted. If omitted, defaults to a year ago.

`organizationID` (`ID`)
The ID of the organization used to filter contributions.

`to` (`DateTime`)
Only contributions made before and up to (including) this time will be counted. If omitted, defaults to the current time or one year from the provided from argument.

`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`email` (`String!`)	The user's publicly visible profile email.
`enterprises` (`EnterpriseConnection`)	A list of enterprises that the user belongs to.

Arguments for `enterprises`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`membershipType` (`EnterpriseMembershipType`)
Filter enterprises returned based on the user's membership type.
The default value is `ALL`.

`orderBy` (`EnterpriseOrder`)
Ordering options for the User's enterprises.

`estimatedNextSponsorsPayoutInCents` (`Int!`)	The estimated next GitHub Sponsors payout for this user/organization in cents (USD).
`followers` (`FollowerConnection!`)	A list of users the given user is followed by.

Arguments for `followers`

`after` (`String`)

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`following` (`FollowingConnection!`)

A list of users the given user is following.

Arguments for `following`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`gist` (`Gist`)

Find gist by repo name.

Arguments for `gist`

`name` (`String!`)
The gist name to find.

`gistComments` (`GistCommentConnection!`)

A list of gist comments made by this user.

Arguments for `gistComments`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`gists` (`GistConnection!`)	A list of the Gists the user has created. **Arguments for `gists`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`GistOrder`) Ordering options for gists returned from the connection. `privacy` (`GistPrivacy`) Filters Gists according to privacy.
`hasSponsorsListing` (`Boolean!`)	True if this user/organization has a GitHub Sponsors listing.
`hovercard` (`Hovercard!`)	The hovercard information for this user in a given context. **Arguments for `hovercard`** `primarySubjectId` (`ID`) The ID of the subject to get the hovercard in the context of.
`interactionAbility` (`RepositoryInteractionAbility`)	The interaction ability settings for this user.
`isBountyHunter` (`Boolean!`)	Whether or not this user is a participant in the GitHub Security Bug Bounty.
`isCampusExpert` (`Boolean!`)	Whether or not this user is a participant in the GitHub Campus Experts Program.
`isDeveloperProgramMember` (`Boolean!`)	Whether or not this user is a GitHub Developer Program member.
`isEmployee` (`Boolean!`)	Whether or not this user is a GitHub employee.
`isFollowingViewer` (`Boolean!`)	Whether or not this user is following the viewer. Inverse of viewerIsFollowing.
`isGitHubStar` (`Boolean!`)	Whether or not this user is a member of the GitHub Stars Program.
`isHireable` (`Boolean!`)	Whether or not the user has marked themselves

	as for hire.
`isSiteAdmin` (`Boolean!`)	Whether or not this user is a site administrator.
`isSponsoredBy` (`Boolean!`)	Whether the given account is sponsoring this user/organization. **Arguments for `isSponsoredBy`** `accountLogin` (`String!`) The target account's login.
`isSponsoringViewer` (`Boolean!`)	True if the viewer is sponsored by this user/organization.
`isViewer` (`Boolean!`)	Whether or not this user is the viewing user.
`issueComments` (`IssueCommentConnection!`)	A list of issue comments made by this user. **Arguments for `issueComments`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list. `orderBy` (`IssueCommentOrder`) Ordering options for issue comments returned from the connection.
`issues` (`IssueConnection!`)	A list of issues associated with this user. **Arguments for `issues`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `filterBy` (`IssueFilters`) Filtering options for issues returned from the connection. `first` (`Int`) Returns the first *n* elements from the list. `labels` (`[String!]`)

`labels` (`[String!]`)
A list of label names to filter the pull requests by.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`IssueOrder`)
Ordering options for issues returned from the connection.

`states` (`[IssueState!]`)
A list of states to filter the issues by.

`itemShowcase` (`ProfileItemShowcase!`)	Showcases a selection of repositories and gists that the profile owner has either curated or that have been selected automatically based on popularity.
`location` (`String`)	The user's public profile location.
`login` (`String!`)	The username used to login.
`monthlyEstimatedSponsorsIncomeInCents` (`Int!`)	The estimated monthly GitHub Sponsors income for this user/organization in cents (USD).
`name` (`String`)	The user's public profile name.
`organization` (`Organization`)	Find an organization by its login that the user belongs to.

Arguments for `organization`

`login` (`String!`)
The login of the organization to find.

`organizationVerifiedDomainEmails` (`[String!]!`)	Verified email addresses that match verified domains for a specified organization the user is a member of.

Arguments for `organizationVerifiedDomainEmails`

`login` (`String!`)
The login of the organization to match verified domains from.

`organizations` (`OrganizationConnection!`)	A list of organizations the user belongs to.

Arguments for `organizations`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`OrganizationOrder`)
Ordering options for the User's organizations.

`packages` (`PackageConnection!`)

A list of packages under the owner.

Arguments for `packages`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`names` (`[String]`)
Find packages by their names.

`orderBy` (`PackageOrder`)
Ordering of the returned packages.

`packageType` (`PackageType`)
Filter registry package by type.

`repositoryId` (`ID`)
Find packages in a repository by ID.

`pinnableItems` (`PinnableItemConnection!`)

A list of repositories and gists this profile owner can pin to their profile.

Arguments for `pinnableItems`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`types` (`[PinnableItemType!]`)
Filter the types of pinnable items that are returned.

`pinnedItems` (`PinnableItemConnection!`)

A list of repositories and gists this profile owner has pinned to their profile.

Arguments for `pinnedItems`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`types` (`[PinnableItemType!]`)
Filter the types of pinned items that are returned.

`pinnedItemsRemaining` (`Int!`)

Returns how many more items this profile owner can pin to their profile.

`project` (`Project`)

Find project by number.

Arguments for `project`

`number` (`Int!`)
The project number to find.

`projectV2` (`ProjectV2`)

Find a project by number.

Arguments for `projectV2`

`number` (`Int!`)
The project number.

`projects` (`ProjectConnection!`)

A list of projects under the owner.

Arguments for `projects`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)

before (String)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ProjectOrder`)
Ordering options for projects returned from the connection.

`search` (`String`)
Query to search projects by, currently only searching by name.

`states` (`[ProjectState!]`)
A list of states to filter the projects by.

`projectsResourcePath` (`URI!`)	The HTTP path listing user's projects.
`projectsUrl` (`URI!`)	The HTTP URL listing user's projects.
`projectsV2` (`ProjectV2Connection!`)	A list of projects under the owner.

Arguments for `projectsV2`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`ProjectV2Order`)
How to order the returned projects.

`query` (`String`)
A project to search for under the the owner.

`pronouns` (`String`)	The user's profile pronouns.
`publicKeys` (`PublicKeyConnection!`)	A list of public keys associated with this user.

Arguments for `publicKeys`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)

Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`pullRequests` (`PullRequestConnection!`)

A list of pull requests associated with this user.

Arguments for `pullRequests`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`baseRefName` (`String`)
The base ref name to filter the pull requests by.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`headRefName` (`String`)
The head ref name to filter the pull requests by.

`labels` (`[String!]`)
A list of label names to filter the pull requests by.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`IssueOrder`)
Ordering options for pull requests returned from the connection.

`states` (`[PullRequestState!]`)
A list of states to filter the pull requests by.

`recentProjects` (`ProjectV2Connection!`)

Recent projects that this user has modified in the context of the owner.

Arguments for `recentProjects`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)

Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`repositories` (`RepositoryConnection!`)

A list of repositories that the user owns.

Arguments for `repositories`

`affiliations` (`[RepositoryAffiliation]`)
Array of viewer's affiliation options for repositories returned from the connection. For example, OWNER will include only repositories that the current viewer owns.

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`hasIssuesEnabled` (`Boolean`)
If non-null, filters repositories according to whether they have issues enabled.

`isArchived` (`Boolean`)
If non-null, filters repositories according to whether they are archived and not maintained.

`isFork` (`Boolean`)
If non-null, filters repositories according to whether they are forks of another repository.

`isLocked` (`Boolean`)
If non-null, filters repositories according to whether they have been locked.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`RepositoryOrder`)
Ordering options for repositories returned from the connection.

`ownerAffiliations`

(`[RepositoryAffiliation]`)
Array of owner's affiliation options for repositories returned from the connection. For example, OWNER will include only repositories that the organization or user being viewed owns.

`privacy` (`RepositoryPrivacy`)
If non-null, filters repositories according to privacy.

`repositoriesContributedTo` (`RepositoryConnection!`)

A list of repositories that the user recently contributed to.

Arguments for `repositoriesContributedTo`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`contributionTypes` (`[RepositoryContributionType]`)
If non-null, include only the specified types of contributions. The GitHub.com UI uses [COMMIT, ISSUE, PULL_REQUEST, REPOSITORY].

`first` (`Int`)
Returns the first *n* elements from the list.

`hasIssues` (`Boolean`)
If non-null, filters repositories according to whether they have issues enabled.

`includeUserRepositories` (`Boolean`)
If true, include user repositories.

`isLocked` (`Boolean`)
If non-null, filters repositories according to whether they have been locked.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`RepositoryOrder`)
Ordering options for repositories returned from the connection.

`privacy` (`RepositoryPrivacy`)
If non-null, filters repositories according to privacy.

`repository` (`Repository`)

Find Repository.

Arguments for `repository`

`followRenames` (`Boolean`)
Follow repository renames. If disabled, a repository referenced by its old name will return an error.
The default value is `true`.

`name` (`String!`)

Name of Repository to find.

`repositoryDiscussionComments` (`DiscussionCommentConnection!`)

Discussion comments this user has authored.

Arguments for `repositoryDiscussionComments`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`onlyAnswers` (`Boolean`)
Filter discussion comments to only those that were marked as the answer.
The default value is `false`.

`repositoryId` (`ID`)
Filter discussion comments to only those in a specific repository.

`repositoryDiscussions` (`DiscussionConnection!`)

Discussions this user has started.

Arguments for `repositoryDiscussions`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`answered` (`Boolean`)
Filter discussions to only those that have been answered or not. Defaults to including both answered and unanswered discussions.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`DiscussionOrder`)
Ordering options for discussions returned from the connection.

`repositoryId` (`ID`)
Filter discussions to only those in a specific

repository.

`states` (`[DiscussionState!]`)
A list of states to filter the discussions by.

`resourcePath` (`URI!`)	The HTTP path for this user.
`savedReplies` (`SavedReplyConnection`)	Replies this user has saved.

Arguments for `savedReplies`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`SavedReplyOrder`)
The field to order saved replies by.

`socialAccounts` (`SocialAccountConnection!`)	The user's social media accounts, ordered as they appear on the user's profile.

Arguments for `socialAccounts`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`sponsoring` (`SponsorConnection!`)	List of users and organizations this entity is sponsoring.

Arguments for `sponsoring`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`SponsorOrder`)
Ordering options for the users and organizations returned from the connection.

`sponsors` (`SponsorConnection!`)

List of sponsors for this user or organization.

Arguments for `sponsors`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`SponsorOrder`)
Ordering options for sponsors returned from the connection.

`tierId` (`ID`)
If given, will filter for sponsors at the given tier. Will only return sponsors whose tier the viewer is permitted to see.

`sponsorsActivities` (`SponsorsActivityConnection!`)

Events involving this sponsorable, such as new sponsorships.

Arguments for `sponsorsActivities`

`actions` (`[SponsorsActivityAction!]`)
Filter activities to only the specified actions.

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)

Returns the first *n* elements from the list.

`includeAsSponsor` (`Boolean`)
Whether to include those events where this sponsorable acted as the sponsor. Defaults to only including events where this sponsorable was the recipient of a sponsorship.
The default value is `false`.

`includePrivate` (`Boolean`)
Whether or not to include private activities in the result set. Defaults to including public and private activities.
The default value is `true`.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`SponsorsActivityOrder`)
Ordering options for activity returned from the connection.

`period` (`SponsorsActivityPeriod`)
Filter activities returned to only those that occurred in the most recent specified time period. Set to ALL to avoid filtering by when the activity occurred. Will be ignored if `since` or `until` is given.
The default value is `MONTH`.

`since` (`DateTime`)
Filter activities to those that occurred on or after this time.

`until` (`DateTime`)
Filter activities to those that occurred before this time.

`sponsorsListing` (`SponsorsListing`)

The GitHub Sponsors listing for this user or organization.

`sponsorshipForViewerAsSponsor` (`Sponsorship`)

The sponsorship from the viewer to this user/organization; that is, the sponsorship where you're the sponsor.

Arguments for `sponsorshipForViewerAsSponsor`

`activeOnly` (`Boolean`)
Whether to return the sponsorship only if it's still active. Pass false to get the viewer's sponsorship back even if it has been

cancelled.
The default value is `true`.

`sponsorshipForViewerAsSponsorable` (`Sponsorship`)

The sponsorship from this user/organization to the viewer; that is, the sponsorship you're receiving.

Arguments for `sponsorshipForViewerAsSponsorable`

`activeOnly` (`Boolean`)
Whether to return the sponsorship only if it's still active. Pass false to get the sponsorship back even if it has been cancelled.
The default value is `true`.

`sponsorshipNewsletters` (`SponsorshipNewsletterConnection!`)

List of sponsorship updates sent from this sponsorable to sponsors.

Arguments for `sponsorshipNewsletters`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`SponsorshipNewsletterOrder`)
Ordering options for sponsorship updates returned from the connection.

`sponsorshipsAsMaintainer` (`SponsorshipConnection!`)

The sponsorships where this user or organization is the maintainer receiving the funds.

Arguments for `sponsorshipsAsMaintainer`

`activeOnly` (`Boolean`)
Whether to include only sponsorships that are active right now, versus all sponsorships this maintainer has ever received.
The default value is `true`.

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`includePrivate` (`Boolean`)
Whether or not to include private sponsorships in the result set.
The default value is `false`.

The default value is `false`.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`SponsorshipOrder`)
Ordering options for sponsorships returned from this connection. If left blank, the sponsorships will be ordered based on relevancy to the viewer.

`sponsorshipsAsSponsor` (`SponsorshipConnection!`)

The sponsorships where this user or organization is the funder.

Arguments for `sponsorshipsAsSponsor`

`activeOnly` (`Boolean`)
Whether to include only sponsorships that are active right now, versus all sponsorships this sponsor has ever made. The default value is `true`.

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`maintainerLogins` (`[String!]`)
Filter sponsorships returned to those for the specified maintainers. That is, the recipient of the sponsorship is a user or organization with one of the given logins.

`orderBy` (`SponsorshipOrder`)
Ordering options for sponsorships returned from this connection. If left blank, the sponsorships will be ordered based on relevancy to the viewer.

`starredRepositories`

(`StarredRepositoryConnection!`)

Repositories the user has starred.

Arguments for `starredRepositories`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`StarOrder`)
Order for connection.

`ownedByViewer` (`Boolean`)
Filters starred repositories to only return repositories owned by the viewer.

`status` (`UserStatus`)	The user's description of what they're currently doing.
`topRepositories` (`RepositoryConnection!`)	Repositories the user has contributed to, ordered by contribution rank, plus repositories the user has created.

Arguments for `topRepositories`

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`RepositoryOrder!`)
Ordering options for repositories returned from the connection.

`since` (`DateTime`)
How far back in time to fetch contributed repositories.

`totalSponsorshipAmountAsSponsorInCents` (`Int`)	The amount in United States cents (e.g., 500 = $5.00 USD) that this entity has spent on GitHub to fund sponsorships. Only returns a value when viewed by the user themselves or by a user who can manage sponsorships for the requested organization.

Arguments for `totalSponsorshipAmountAsSponsorInCents`

`since` (`DateTime`)
Filter payments to those that occurred on or after this time.

`sponsorableLogins` (`[String!]`)

`sponsorableLogins` (`[String!]`)
Filter payments to those made to the users or organizations with the specified usernames.

`until` (`DateTime`)
Filter payments to those that occurred before this time.

`twitterUsername` (`String`)	The user's Twitter username.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this user.
`viewerCanChangePinnedItems` (`Boolean!`)	Can the viewer pin repositories and gists to the profile?.
`viewerCanCreateProjects` (`Boolean!`)	Can the current viewer create new projects on this owner.
`viewerCanFollow` (`Boolean!`)	Whether or not the viewer is able to follow the user.
`viewerCanSponsor` (`Boolean!`)	Whether or not the viewer is able to sponsor this user/organization.
`viewerIsFollowing` (`Boolean!`)	Whether or not this user is followed by the viewer. Inverse of isFollowingViewer.
`viewerIsSponsoring` (`Boolean!`)	True if the viewer is sponsoring this user/organization.
`watching` (`RepositoryConnection!`)	A list of repositories the given user is watching.

Arguments for `watching`

`affiliations` (`[RepositoryAffiliation]`)
Affiliation options for repositories returned from the connection. If none specified, the results will include repositories for which the current viewer is an owner or collaborator, or member.

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`hasIssuesEnabled` (`Boolean`)
If non-null, filters repositories according to whether they have issues enabled.

`isLocked` (`Boolean`)
If non-null, filters repositories according to whether they have been locked.

whether they have been locked.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`RepositoryOrder`)
Ordering options for repositories returned from the connection.

`ownerAffiliations` (`[RepositoryAffiliation]`)
Array of owner's affiliation options for repositories returned from the connection. For example, OWNER will include only repositories that the organization or user being viewed owns.

`privacy` (`RepositoryPrivacy`)
If non-null, filters repositories according to privacy.

`websiteUrl` (`URI`)	A URL pointing to the user's public website/blog.

UserBlockedEvent

Represents a `user_blocked` event on a given user.

`UserBlockedEvent` Implements

- `Node`

Fields for `UserBlockedEvent`

Name	Description
`actor` (`Actor`)	Identifies the actor who performed the event.
`blockDuration` (`UserBlockDuration!`)	Number of days that the user was blocked for.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`subject` (`User`)	The user who was blocked.

UserConnection

The connection type for User.

Fields for `UserConnection`

Name	Description
`edges` (`[UserEdge]`)	A list of edges.
`nodes` (`[User]`)	A list of nodes.

`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

UserContentEdit

An edit on user content.

`UserContentEdit` Implements

- `Node`

Fields for `UserContentEdit`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`deletedAt` (`DateTime`)	Identifies the date and time when the object was deleted.
`deletedBy` (`Actor`)	The actor who deleted this content.
`diff` (`String`)	A summary of the changes for this edit.
`editedAt` (`DateTime!`)	When this content was edited.
`editor` (`Actor`)	The actor who edited this content.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.

UserContentEditConnection

A list of edits to content.

Fields for `UserContentEditConnection`

Name	Description
`edges` (`[UserContentEditEdge]`)	A list of edges.
`nodes` (`[UserContentEdit]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

UserContentEditEdge

An edge in a connection.

Fields for `UserContentEditEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`UserContentEdit`)	The item at the end of the edge.

UserEdge

Represents a user.

Fields for `UserEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`User`)	The item at the end of the edge.

UserEmailMetadata

Email attributes from External Identity.

Fields for `UserEmailMetadata`

Name	Description
`primary` (`Boolean`)	Boolean to identify primary emails.
`type` (`String`)	Type of email.
`value` (`String!`)	Email id.

UserStatus

The user's description of what they're currently doing.

`UserStatus` Implements

- `Node`

Fields for `UserStatus`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`emoji` (`String`)	An emoji summarizing the user's status.
`emojiHTML` (`HTML`)	The status emoji as HTML.

`expiresAt` (`DateTime`)	If set, the status will not be shown after this date.
`indicatesLimitedAvailability` (`Boolean!`)	Whether this status indicates the user is not fully available on GitHub.
`message` (`String`)	A brief message describing what the user is doing.
`organization` (`Organization`)	The organization whose members can see this status. If null, this status is publicly visible.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`user` (`User!`)	The user who has this status.

UserStatusConnection

The connection type for UserStatus.

Fields for `UserStatusConnection`

Name	Description
`edges` (`[UserStatusEdge]`)	A list of edges.
`nodes` (`[UserStatus]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

UserStatusEdge

An edge in a connection.

Fields for `UserStatusEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`UserStatus`)	The item at the end of the edge.

VerifiableDomain

A domain that can be verified or approved for an organization or an enterprise.

`VerifiableDomain` Implements

- `Node`

Fields for `VerifiableDomain`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`dnsHostName` (`URI`)	The DNS host name that should be used for verification.
`domain` (`URI!`)	The unicode encoded domain.
`hasFoundHostName` (`Boolean!`)	Whether a TXT record for verification with the expected host name was found.
`hasFoundVerificationToken` (`Boolean!`)	Whether a TXT record for verification with the expected verification token was found.
`isApproved` (`Boolean!`)	Whether or not the domain is approved.
`isRequiredForPolicyEnforcement` (`Boolean!`)	Whether this domain is required to exist for an organization or enterprise policy to be enforced.
`isVerified` (`Boolean!`)	Whether or not the domain is verified.
`owner` (`VerifiableDomainOwner!`)	The owner of the domain.
`punycodeEncodedDomain` (`URI!`)	The punycode encoded domain.
`tokenExpirationTime` (`DateTime`)	The time that the current verification token will expire.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`verificationToken` (`String`)	The current verification token for the domain.

VerifiableDomainConnection

The connection type for VerifiableDomain.

Fields for `VerifiableDomainConnection`

Name	Description
`edges` (`[VerifiableDomainEdge]`)	A list of edges.
`nodes` (`[VerifiableDomain]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

VerifiableDomainEdge

An edge in a connection.

Fields for `VerifiableDomainEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`VerifiableDomain`)	The item at the end of the edge.

ViewerHovercardContext

A hovercard context with a message describing how the viewer is related.

`ViewerHovercardContext` Implements

- `HovercardContext`

Fields for `ViewerHovercardContext`

Name	Description
`message` (`String!`)	A string describing this context.
`octicon` (`String!`)	An octicon to accompany this context.
`viewer` (`User!`)	Identifies the user who is related to this context.

Workflow

A workflow contains meta information about an Actions workflow file.

`Workflow` Implements

- `Node`
- `UniformResourceLocatable`

Fields for `Workflow`

Name	Description
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`name` (`String!`)	The name of the workflow.
`resourcePath` (`URI!`)	The HTTP path for this workflow.
`runs` (`WorkflowRunConnection!`)	The runs of the workflow.

Arguments for `runs`

`after` (`String`)

`after` (`String`)
Returns the elements in the list that come after the specified cursor.

`before` (`String`)
Returns the elements in the list that come before the specified cursor.

`first` (`Int`)
Returns the first *n* elements from the list.

`last` (`Int`)
Returns the last *n* elements from the list.

`orderBy` (`WorkflowRunOrder`)
Ordering options for the connection.

`state` (`WorkflowState!`)	The state of the workflow.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this workflow.

WorkflowFileReference

A workflow that must run for this rule to pass.

Fields for `WorkflowFileReference`

Name	Description
`path` (`String!`)	The path to the workflow file.
`ref` (`String`)	The ref (branch or tag) of the workflow file to use.
`repositoryId` (`Int!`)	The ID of the repository where the workflow is defined.
`sha` (`String`)	The commit SHA of the workflow file to use.

WorkflowRun

A workflow run.

`WorkflowRun` Implements

- `Node`
- `UniformResourceLocatable`

Fields for `WorkflowRun`

Name	Description

`checkSuite` (`CheckSuite!`)	The check suite this workflow run belongs to.
`createdAt` (`DateTime!`)	Identifies the date and time when the object was created.
`databaseId` (`Int`)	Identifies the primary key from the database.
`deploymentReviews` (`DeploymentReviewConnection!`)	The log of deployment reviews. **Arguments for `deploymentReviews`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`event` (`String!`)	The event that triggered the workflow run.
`file` (`WorkflowRunFile`)	The workflow file.
`pendingDeploymentRequests` (`DeploymentRequestConnection!`)	The pending deployment requests of all check runs in this workflow run. **Arguments for `pendingDeploymentRequests`** `after` (`String`) Returns the elements in the list that come after the specified cursor. `before` (`String`) Returns the elements in the list that come before the specified cursor. `first` (`Int`) Returns the first *n* elements from the list. `last` (`Int`) Returns the last *n* elements from the list.
`resourcePath` (`URI!`)	The HTTP path for this workflow run.
`runNumber` (`Int!`)	A number that uniquely identifies this workflow run in its parent workflow.
`updatedAt` (`DateTime!`)	Identifies the date and time when the object was last updated.
`url` (`URI!`)	The HTTP URL for this workflow run.
`workflow` (`Workflow!`)	The workflow executed in this workflow run.

WorkflowRunConnection

The connection type for WorkflowRun.

Fields for `WorkflowRunConnection`

Name	Description
`edges` (`[WorkflowRunEdge]`)	A list of edges.
`nodes` (`[WorkflowRun]`)	A list of nodes.
`pageInfo` (`PageInfo!`)	Information to aid in pagination.
`totalCount` (`Int!`)	Identifies the total count of items in the connection.

WorkflowRunEdge

An edge in a connection.

Fields for `WorkflowRunEdge`

Name	Description
`cursor` (`String!`)	A cursor for use in pagination.
`node` (`WorkflowRun`)	The item at the end of the edge.

WorkflowRunFile

An executed workflow file for a workflow run.

`WorkflowRunFile` Implements

- `Node`
- `UniformResourceLocatable`

Fields for `WorkflowRunFile`

Name	Description
`path` (`String!`)	The path of the workflow file relative to its repository.
`repositoryFileUrl` (`URI!`)	The direct link to the file in the repository which stores the workflow file.
`repositoryName` (`URI!`)	The repository name and owner which stores the workflow file.
`resourcePath` (`URI!`)	The HTTP path for this workflow run file.

`run` (`WorkflowRun!`)	The parent workflow run execution for this file.
`url` (`URI!`)	The HTTP URL for this workflow run file.
`viewerCanPushRepository` (`Boolean!`)	If the viewer has permissions to push to the repository which stores the workflow.
`viewerCanReadRepository` (`Boolean!`)	If the viewer has permissions to read the repository which stores the workflow.

WorkflowsParameters

Require all changes made to a targeted branch to pass the specified workflows before they can be merged.

Fields for `WorkflowsParameters`

Name	Description
`workflows` (`[WorkflowFileReference!]!`)	Workflows that must pass for this rule to pass.