Table of contents

ATLASSIAN CLOUD

Atlassian Cloud architecture and operational practices

Learn more about the Atlassian Cloud architecture and the operational practices we use

Security controls

Introduction

Atlassian cloud apps and data are hosted on industry-leading cloud provider Amazon Web Services (AWS). Our products run on a platform as a service (PaaS) environment that is split into two main sets of infrastructure that we refer to as Micros and non-Micros. Jira, Confluence, Jira Product Discovery, Statuspage, Guard, and Bitbucket run on the Micros platform, while Opsgenie and Trello run on the non-Micros platform.

Cloud infrastructure

provider and its highly available data center facilities in multiple regions worldwide. Each AWS region is a separate geographical location with multiple, isolated, and physically-separated groups of data centers known as Availability Zones (AZs).

We leverage AWS' compute, storage, network, and data services to build our products and platform components, which enables us to utilize redundancy capabilities offered by AWS, such as availability zones and regions.

Availability zones

Each Availability zone is designed to be isolated from failures in the other zones and to provide inexpensive, low-latency network connectivity to other AZs in the same region. This multi-zone high availability is the first line of defense for geographic and environmental risks and means that services running in multi-AZ deployments should be able to withstand AZ failure.

Jira and Confluence use the multi-AZ deployment mode for Amazon RDS (Amazon Relational Database Service). In a multi-AZ deployment, Amazon RDS provisions and maintains a synchronous standby replica in a different AZ within the same region to provide redundancy and failover capability. The AZ failover is automated and typically takes 60-120 seconds, so that database operations can resume as quickly as possible without administrative intervention. Opsgenie, Statuspage, Trello, and Jira Align use similar deployment strategies, with small variances in replica timing and failover timing.

Data location

Table of contents

Cloud infrastructure

Cloud platform architecture

Security controls

Table of contents

Bitbucket data is located in two different availability zones within the US-East region.

However, we understand that some of you may require that your data remain in a specific location; therefore we offer data residency options. Currently, data residency is available for Jira, Jira Service Management, Jira Product Discovery, and Confluence across 11 regions, including the US, EU, UK, Australia, Canada, Germany, India, Japan, Singapore, South Korea, and Switzerland. Read our documentation to learn more about data residency and the relevant in-scope product data. Additionally, you can follow our roadmap for updates on data residency, including expansions to new products, regions, and data types.

Data backups

We operate a comprehensive backup program at Atlassian. This includes our internal systems, where our backup measures are designed in line with system recovery requirements. With respect to our cloud apps, and specifically referring to you and your application data, we also have extensive backup measures in place. We use the snapshot feature of Amazon RDS (Relational database service) to create automated daily backups of each RDS instance.

Amazon RDS snapshots are retained for 30 days with support for point-in time recovery and are encrypted using AES-256 encryption. Backup data is not stored offsite but is replicated to multiple data centers within a particular AWS region. We also perform quarterly testing of our backups.

For Bitbucket, storage snapshots are retained for 7 days with support for point-in-time recovery.

Table of contents

Cloud infrastructure

Cloud platform architecture

Security controls

scripts, or deleted work items, projects, or sites. To avoid data loss, we recommend making regular backups. Learn more about creating backups in the support documentation for your product.

Data center security

AWS maintains multiple certifications for the protection of their data centers. These certifications address physical and environmental security, system availability, network and IP backbone access, customer provisioning and problem management. Access to the data centers is limited to authorized personnel only, as verified by biometric identity verification measures. Physical security measures include: on-premises security guards, closed circuit video monitoring, man traps, and additional intrusion protection measures.

Cloud platform architecture

Distributed services architecture

With this AWS architecture, we host a number of platform and product services that are used across our solutions. This includes platform capabilities that are shared and consumed across multiple Atlassian products, such as Media, Identity, and Commerce, experiences such as our Editor, and product-specific capabilities, like Jira Work Item service and Confluence Analytics.

Table of contents

Cloud infrastructure

Cloud platform architecture

Security controls

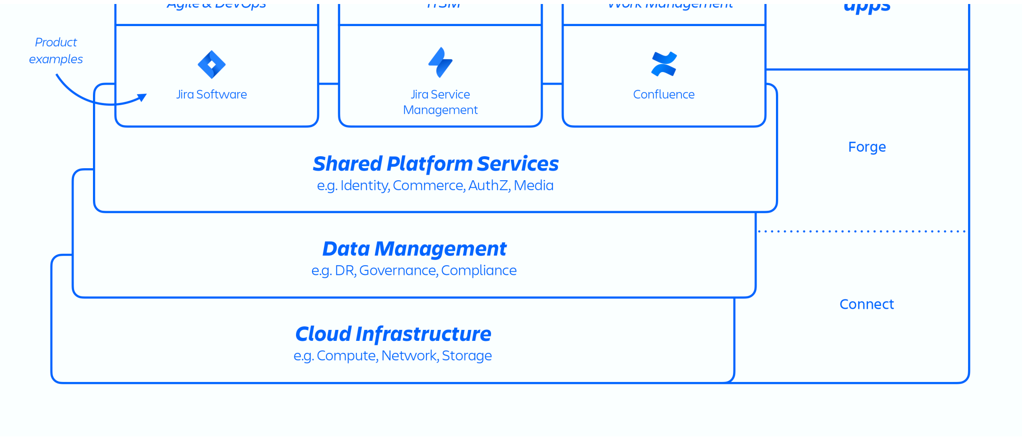

Figure 1

Atlassian developers provision these services either through kubernetes or using an internally developed platform-as-a-service (PaaS), called Micros, both of which automatically orchestrates the deployment of shared services, infrastructure, data stores, and their management capabilities, including security and compliance control requirements (see figure 1 above). Typically, an Atlassian product consists of multiple "containerized" services that are deployed on AWS using Micros or kubernetes. Atlassian products use core platform capabilities (see figure 2 below) that range from request routing to binary object stores, authentication/authorization, transactional user-generated content (UGC) and entity relationships stores, data lakes, common logging, request tracing, observability, and analytical services. These micro-services are built using approved technical stacks standardized at the platform level:

Table of contents

Cloud infrastructure

Cloud platform architecture

Security controls

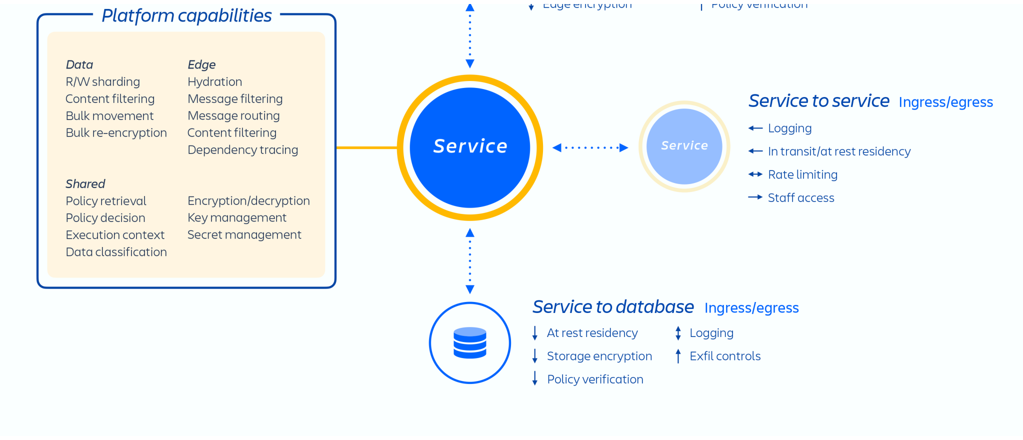

Figure 2

Multi-tenant architecture

On top of our cloud infrastructure, we built and operate a multi-tenant micro-service architecture along with a shared platform that supports our products. In a multi-tenant architecture, a single service serves multiple customers, including databases and compute instances required to run our cloud apps. Each shard (essentially a container – see figure 3 below) contains the data for multiple tenants, but each tenant's data is isolated and inaccessible to other tenants. It is important to note that we do not offer a single tenant architecture.

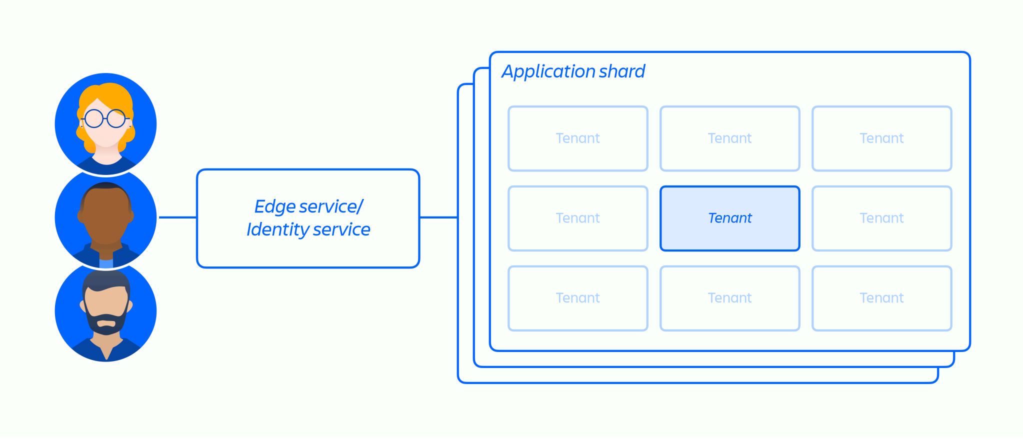

Figure 3

Our microservices are built with least privilege in mind and designed to minimize the scope of any zero-day

Table of contents

Cloud infrastructure

Cloud platform architecture

Security controls

has its own data storage that can only be accessed with the authentication protocol for that specific service, which means that no other service has read or write access to that API.

We've focused on isolating microservices and data, rather than providing dedicated per-tenant infrastructure because it narrows the access to a single system's narrow purview of data across many customers. Because the logic has been decoupled and data authentication and authorization occurs at the application layer, this acts as an additional security check as requests are sent to these services. Thus, if a microservice is compromised, it will only result in limited access to the data a particular service requires.

Tenant provisioning and lifecycle

When a new customer is provisioned, a series of events trigger the orchestration of distributed services and provisioning of data stores. These events can be generally mapped to one of seven steps in the lifecycle:

1. Commerce systems are immediately updated with the latest metadata and access control information for that customer, and then a provisioning orchestration system aligns the "state of the provisioned resources" with the license state through a series of tenant and product events.

Tenant events

These events affect the tenant as a whole and can either be:

- Creation: a tenant is created and used for brand new sites
- Destruction: an entire tenant is deleted

Product events

- Deactivation: after the de-activation of certain products or apps
- Suspension: after the suspension of a given existing product, thus disabling access to a given site that they own
- Un-suspension: after the un-suspension of a given existing product, thus enabling access to a site that they own
- License update: contains information regarding the number of license seats for a given product as well as its status (active/inactive)

2. Creation of the customer site and activation of the correct set of products for the customer. The concept of a site is the container of multiple products licensed to a particular customer. (e.g. Confluence and Jira Software for site-name.atlassian.net).

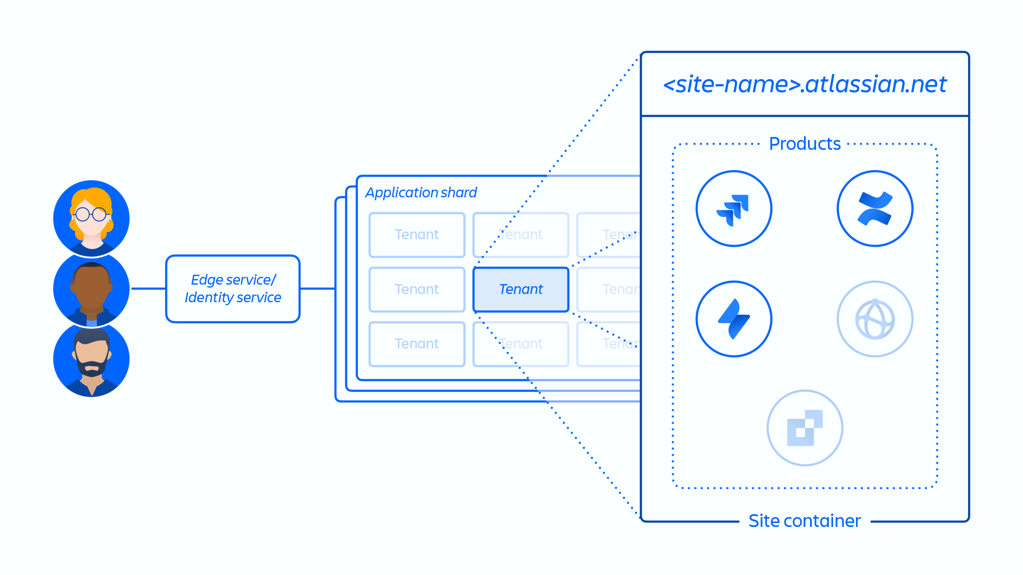

Figure 4

3. Provisioning of products within the customer site in the designated region.

When a product is provisioned it will have the majority of its content hosted close to where users are accessing it. To data between regions as needed.

For some of our products, we also offer data residency. Data residency allows customers to choose whether product data is globally distributed or held in place in one of our defined geographic locations.

4. Creation and storage of the customer site and product(s) core metadata and configuration.

5. Creation and storage of the site and product(s) identity data, such as users, groups, permissions, etc.

6. Provisioning of product databases within a site, e.g. Jira family of products, Confluence, Compass, Atlas.

7. Provisioning of the product(s) licensed apps.

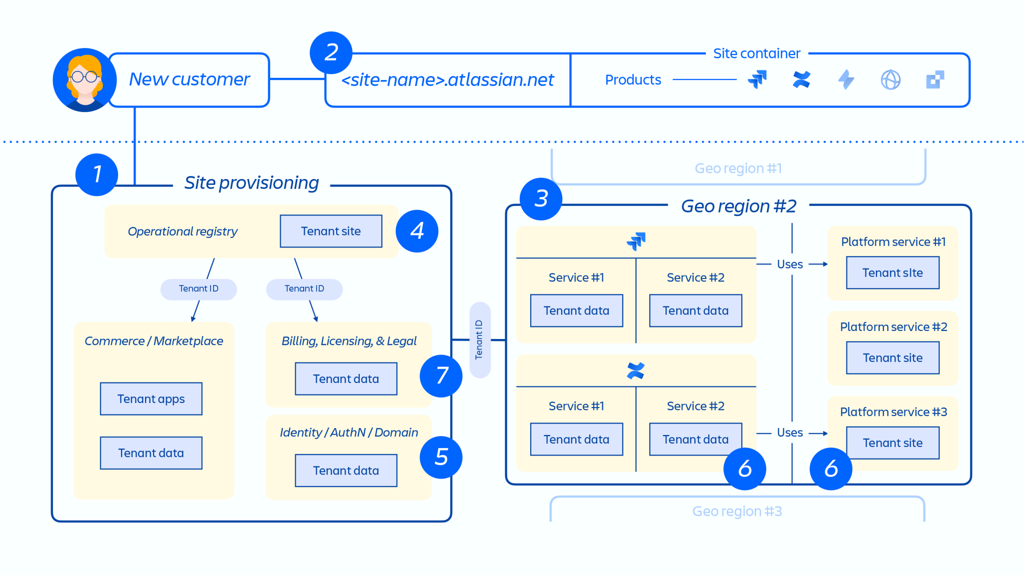

Figure 5

Figure 5 above demonstrates how a customer's site is deployed across our distributed architecture, not just in a single database or store. This includes multiple physical and logical locations that store meta-data, configuration data, product data, platform data and other related site info.

infrastructure when using our cloud apps, we have measures in place to ensure they are logically separated so that the actions of one customer cannot compromise the data or service of other customers.

Atlassian's approach to achieving this varies across our applications. In the case of Jira and Confluence Cloud, we use a concept we refer to as the "tenant context" to achieve logical isolation of our customers. This is implemented both in the application code, and managed by something we have built called the tenant context service (TCS). This concept ensures that:

- Each customer's data is kept logically segregated from other tenants when at-rest
- Any requests that are processed by Jira or Confluence have a tenant-specific view so other tenants are not impacted

In broad terms, the TCS works by storing a context for individual customer tenants. The context for each tenant is associated with a unique ID stored centrally by the TCS, and includes a range of metadata associated with that tenant, such as which databases the tenant is in, what licenses the tenant has, what features they can access, and a range of other configuration information. When a customer accesses Jira or Confluence cloud, the TCS uses the tenant ID to collate that metadata, which is then linked with any operations the tenant undertakes in the application throughout their session.

Atlassian edges

Your data is also safeguarded through something that we call an edge - virtual walls that we build around our software. When a request comes in, it is sent to the nearest

- They land on the Atlassian edge closest to the user. The edge will verify the user's session and identity through your identity system.
- The edge determines where your product data is located, based on data in the TCS information.
- The edge forwards the request to the target region, where it lands on a compute node.
- The node uses the tenant configuration system to determine information, such as the license and database location, and calls out to various other data stores and services (e.g. the Media platform that hosts images and attachments) to retrieve the information required to service the request.
- The original user request with information assembled from its previous calls to other services.

Security controls

Because our cloud apps leverage a multi-tenant architecture, we can layer additional security controls into the decoupled application logic. A per-tenant monolithic application wouldn't typically introduce further authorization checks or rate limiting, for example, on a high volume of queries or exports. The impact of a single zero-day is dramatically reduced as the scope of services are narrowed.

In addition, we've built additional preventative controls into our products that are fully hosted on our Atlassian platform. The primary preventative controls include:

- Key management
- Data encryption

Table of contents

Cloud infrastructure

Cloud platform architecture

Security controls

Service authentication and authorization

Our platform uses a least privilege model for accessing data. This means that all data is restricted to only the service responsible for saving, processing, or retrieving it. For example, the media services, which allows you to have a consistent file upload and download experience across our cloud apps, have dedicated storage provisioned that no other services at Atlassian can access. Any service that requires access to the media content needs to interact with the media services API. As a result, strong authentication and authorization at the service layer also enforces strong separation of duties and least privilege access to data.

We use JSON web tokens (JWTs) to ensure signing authority outside of the application, so our identity systems and tenant context are the source of truth. Tokens can’t be used for anything other than what they are authorized for. When you or someone on your team makes a call to a microservice or shard, the tokens are passed to your identity system and validated against it. This process ensures that the token is current and signed before sharing the appropriate data. When combined with the authorization and authentication required to access these microservices, if a service is compromised, it’s limited in scope.

However, we know that sometimes identity systems can be compromised. To mitigate this risk, we use two mechanisms. First, TCS and the identity proxies are highly replicated. We have a TCS sidecar for almost every microservice and we use proxy sidecars that offshoot to the more, we can pick up on that quickly and remediate the issue.

In addition, we don't wait for someone to find a vulnerability in our products or platform. We're actively identifying these scenarios so there is minimal impact to you and we run a number of security programs to identify, detect, and respond to security threats.

Tenant context service

We ensure that requests to any microservices contain metadata about the customer - or tenant - that is requesting access. This is called the tenant context service. It's populated directly from our provisioning systems. When a request is started, the context is read and internalized in the running service code, which is used to authorize the user. All service access, and thus data access, in Jira and Confluence require this tenant context or the request will be rejected.

Service authentication and authorization is applied through Atlassian service authentication protocol (ASAP). An explicit allowlist determines which services may communicate, and authorization details specify which commands and paths are available. This limits potential lateral movement of a compromised service.

Service authentication and authorization, as well as egress, are controlled by a set of dedicated proxies. This removes the ability for application code vulnerabilities to impact these controls. Remote code execution would require compromising the underlying host and bypassing the Docker container boundaries - not just the ability to modify application logic. Rather, our host level intrusion detection flags discrepancies.

Table of contents

Cloud infrastructure

Cloud platform architecture

Security controls

emit webhooks or communicate to other microservices that are prohibited from doing so.

Data encryption

Customer data in Atlassian cloud apps is encrypted during transmission utilizing TLS 1.2 or higher, incorporating perfect forward secrecy (PFS) to safeguard against unauthorized information disclosure and data modification. We adhere to NIST-recommended TLS 1.2+ protocols, which enforce the use of strong ciphers and key lengths as supported by the browser.

Customer data, including attachments, stored on the cloud services such as Jira Software Cloud, Jira Service Management Cloud, Jira, Bitbucket Cloud, Confluence Cloud, Statuspage, Opsgenie, and Trello are protected using industry-standard AES-256 encryption at rest.

Personally Identifiable Information (PII) transmission is protected through encryption and robust data access controls, which are designed to ensure that data is securely transmitted to its intended destination. Atlassian's Cryptography and Encryption Policy outlines principles for implementing encryption and cryptography to mitigate risks related to storing and transmitting PII. The encryption algorithms for protecting PII are aligned with the classification level of the PII, as specified by Atlassian's internal Data Security & Information Lifecycle Management policies. This ensures that sensitive data is adequately secured based on its classification. To learn more about how we collect, share, and use customer data, refer to our privacy page.

To keep up to date on additional data encryption capabilities, see our cloud roadmap.

Table of contents

Cloud infrastructure

Cloud platform architecture

Security controls

(KMS) for managing cryptographic keys used for data encryption and decryption. By design, these KMS keys are backed by key materials secured in hardware security modules (HSMs) that are validated by the NIST Cryptographic Module Validation Program. The secure-by-design approach of AWS KMS with FIPS-validated HSMs enables defense in depth where key management is concerned. This prevents both AWS and Atlassian employees from retrieving plaintext key materials in KMS or the HSMs.

Envelope encryption is applied to data-in-transit and data-at-rest. Data keys are created corresponding to each service, and only the authorized services are allowed to encrypt or decrypt in an implicit-deny fashion. Data keys are then enveloped (encrypted by the corresponding KMS CMK resources) for protection.

Volume or disk-level encryption is implemented as necessary, particularly for resources like databases and object stores that are directly managed through AWS-managed services. The cryptographic keys used for this encryption are provisioned and safeguarded by the same HSM sources.

Both KMS keys and data keys are periodically rotated to minimize potential attack surfaces. When a KMS key is rotated to a new version, the existing data keys that were encrypted using the old or previous versions of the KMS keys can only be decrypted by the old KMS keys. Meanwhile, any new data keys created after the KMS key rotation will be encrypted and decrypted using the new, active version of the KMS key. The management of data key rotation is governed by usage limits, which can be specified in terms of maximum operations or maximum time-to-live (TTL). For

Multi-region KMSs and secure key caches are implemented to achieve high availability and a desired performance level.

For additional details, read this blog.

Bring-your-own key (BYOK)

For greater control over your product data, Atlassian cloud supports bring-your-own-key (BYOK) encryption capability for a selected and growing product data portfolio. Learn more about BYOK here.

Atlassian BYOK encryption does not introduce performance overhead or negatively impact user experience due to the efficient, secure caching mechanism employed by Atlassian systems.

RESOURCES

Technical support

Purchasing & licensing

Atlassian Community

Knowledge base

Marketplace

My account

Create support ticket →

LEARN

Partners

Training & certification

Documentation

Developer resources

Enterprise services

See all resources →

Copyright © 2025 Atlassian

Privacy Policy Terms Impressum

English ▼

Multi-tenant architecture

Tenant provisioning and lifecycle

Tenant separation

Atlassian edges

Security controls

Service authentication and authorization

Tenant context service

Data encryption

Cryptographic key management

- Bring-your-own key (BYOK)