NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Introduction to DevOps

Cloud Infrastructure Engineering

Nanyang Technological University
& Skills Union - 2022/2023

Course Content

- Self Study Check In
- Recap about Agile Methodology (2.8 Software Deployment Req.)
- DevOps
- AWS DevOps
- Activity

Q1: Which one of the following techniques makes DevOps a successful methodology to develop and deliver software?

A: DevOps enables you to organize your teams around your organizational mission
B: DevOps enables you to create your software with quality and monitoring
C: DevOps enables you to quickly identify, fix and learn from errors
D: All of the above

Q2: Which of the following statement best describes the goal of DevOps?

A. Establish an environment to release more reliable applications faster
B.Establish an environment where the release of applications is valued more than its quality
C.Establish an environment where application development team performs all the operation tasks
D.All of the above

Q3: What are some of the common questions usually asked during daily standup / huddle?

Q4: Explain the relation between Agile Methodology with DevOps.

Recap about Agile Methodology (2.8 Software Deployment Req.)

What is Agile Methodology?

Agile Methodology meaning a practice that promotes continuous iteration of development and testing throughout the software development life cycle of the project.

In the Agile model in software testing, both development and testing activities are concurrent, unlike the Waterfall model.

Activity

Instructor

- Ask to use AWS use single region for all learner for easier monitoring

DevOps

A compound of development (Dev) and operations (Ops), DevOps is the union of people, process, and technology to continually provide value to customers.

What does DevOps mean for teams? DevOps enables formerly siloed roles—development, IT operations, quality engineering, and security—to coordinate and collaborate to produce better, more reliable products. By adopting a DevOps culture along with DevOps practices and tools, teams gain the ability to better respond to customer needs, increase confidence in the applications they build, and achieve business goals faster.

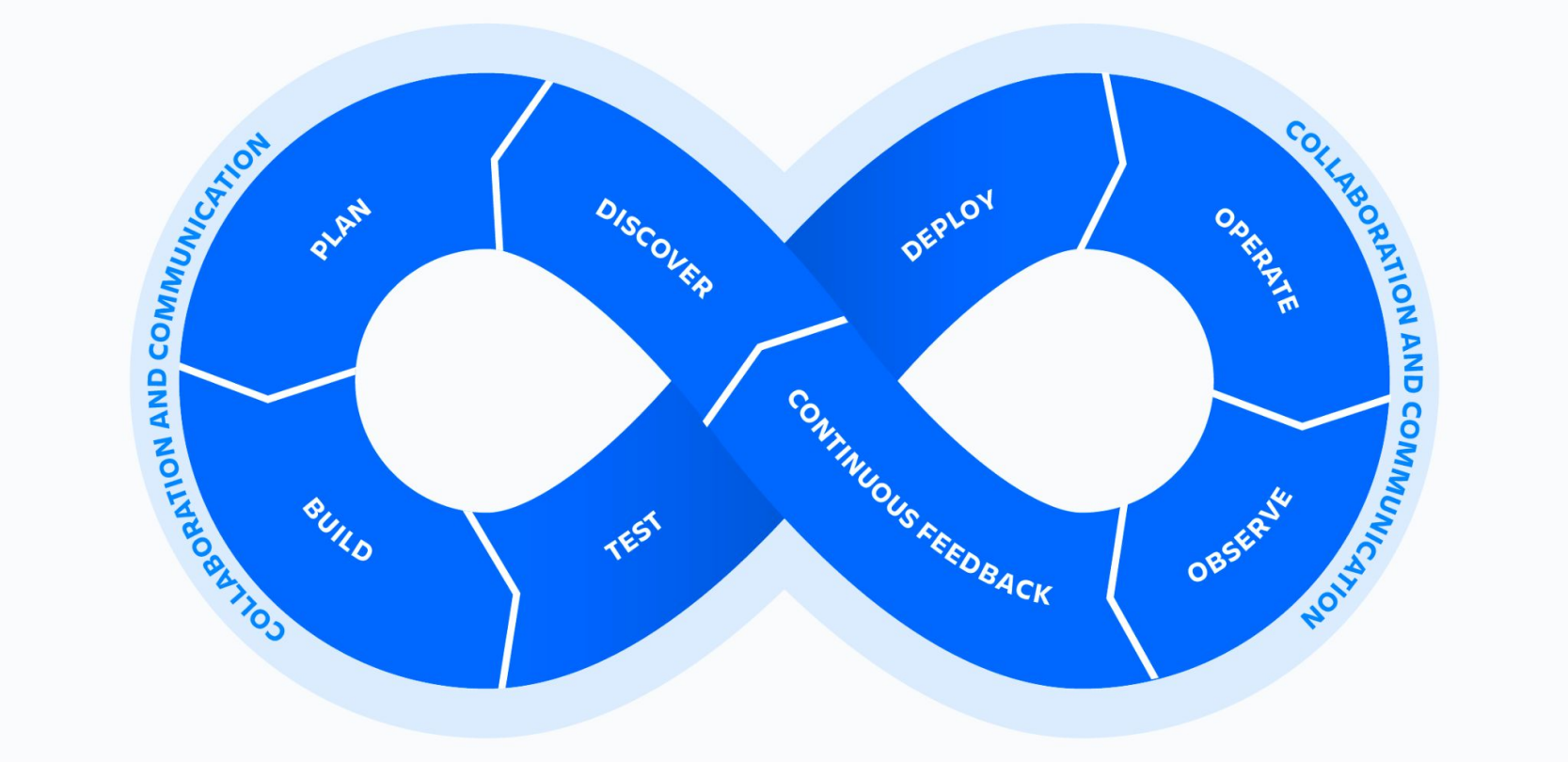
How does DevOps work?

DevOps is a methodology that emphasizes **collaboration and communication between development and operations** teams throughout the entire product lifecycle. This approach aims to improve the speed and quality of software deployment by breaking down silos between development and operations teams, and by using tools to automate and accelerate processes.

DevOps teams use a set of tools, commonly referred to as a toolchain, to **automate and streamline processes**. These tools help teams to implement key DevOps practices such as continuous integration, continuous delivery, automation, and collaboration.

The DevOps approach can also be applied to teams outside of development, such as security teams. This is known as DevSecOps, in which security becomes an integrated and active part of the development process.

DevOps lifecycle



The DevOps lifecycle

There may be variations in the number and names of workflows depending on the organization, but generally, these six stages are included:

1. Planning - This stage involves scoping out new features and functionality for the next release, based on feedback from end-users and other internal stakeholders.
2. Development - In this stage, developers test, code, and build new and enhanced features, based on user stories and work items from the backlog.
3. Integration - This stage involves integrating the new code into the existing codebase, testing, and packaging it into an executable for deployment.
4. Deployment - This stage is where the runtime build output is deployed to a runtime environment, usually a development environment, where runtime tests are executed for quality, compliance, and security.
5. Operations - This stage involves monitoring the performance, behavior, and availability of the deployed features, to ensure they provide value to end-users.
6. Learning - This stage involves gathering feedback from end-users and customers on features, functionality, performance, and business value, to take back to the planning stage for future enhancements and features.

Agile VS DevOps

DevOps	Agile
A technical practice to achieve the end solution / product	It is more about processes such as scrum methodologies and sprints
Operations as Deployment is a small part of ops. Other being configuration management etc.	It is the link between design thinking and DevOps
Aligns Ops with stability such as scripting deployment and standardizing app build	Aligns development with business priorities, thus focusing on speeding up the development process
DevOps help to bridge the line between multiple cross-functional teams(Development, QA, Monitoring, Deployment, Management etc.) by simplifying the process	Agile help to bridge the line between these teams through daily stand ups where roadblocks are discussed and resolved.
Aims at faster deliveries through automation and configuration management tools	Aims at streamlined delivery via tools such as Kanban boards, that help in project management and collaboration

How DevOps and agile work together?

Both DevOps and agile offer a structure and framework that can speed software delivery. You do not need to choose between DevOps or agile—instead, you can make use of both methodologies. Agile is strong on methods to organize work, such as through Scrum or Kanban, and DevOps drives a broader culture of getting software delivered faster and more reliably.

Instead of a decision of DevOps versus agile, the question—actually—is how to practice both. When considering how to build a development practice with the best of DevOps and agile, here are some examples of key benefits and features that can help you create a highly optimized development environment.

Additional - DevSecOps

Additional - DevSecOps

DevSecOps (Development, Security, and Operations) is a software development approach that integrates security practices into the entire software development lifecycle (SDLC) from design to deployment. It emphasizes collaboration and communication between development, security, and operations teams in order to ensure that security is an integral part of the software development process.

DevSecOps aims to integrate security into the CICD pipeline, by automating security testing, vulnerability scanning, and compliance checking, as well as security-related code reviews, so that security issues can be identified and addressed early in the development process. It also includes continuous monitoring and threat detection, to ensure that security issues are identified and resolved as quickly as possible.

DevSecOps also promotes a culture of collaboration, communication, and shared responsibility for security among development, security, and operations teams. This helps to ensure that security considerations are integrated into the development process, and that the resulting software is secure and compliant.

DevSecOps enables teams to deliver software faster and more frequently, while maintaining a high level of security and compliance. It also helps to identify and mitigate security risks early in the development process, reducing the chance of a security incident occurring in production.

AWS DevOps Best Practices

- Use version control: Use a version control system such as Git to manage and track changes to your code. This makes it easier to collaborate with others and roll back changes if necessary.
- Automate your infrastructure: Use AWS CloudFormation or AWS Elastic Beanstalk to automate the provisioning and management of your infrastructure. This allows you to quickly and easily deploy new resources or roll back changes.
- Use CodePipeline and CodeBuild: Use AWS CodePipeline and CodeBuild to automate the process of building, testing, and deploying your code. This allows you to quickly and easily deploy new features and updates to your customers.
- Monitor your resources: Use AWS CloudWatch to monitor the performance of your resources and to receive notifications when certain thresholds are breached.
- Enable security and compliance: Use AWS services such as IAM and Config to enable security and compliance for your resources.
- Use CloudTrail and CloudWatch: Use AWS CloudTrail to log all of your AWS account activity and use CloudWatch to monitor the logs.
- Use AWS CodeStar: Use AWS CodeStar to create a comprehensive and easy-to-use DevOps toolchain for your application.
- Implement disaster recovery: Use AWS services such as S3 and RDS to implement disaster recovery for your application.
- Use AWS Elastic Beanstalk for Deployment: Use AWS Elastic Beanstalk for easy deployment of your applications, it provides a platform for deploying web apps and services, handling capacity provisioning, load balancing, and automatic scaling.
- Continuously measure, monitor, and improve: Continuously measure the performance of your system, monitor its availability and scalability, and continuously improve it using the feedback collected by monitoring.

CI/CD

CICD Pipeline

CICD (Continuous Integration and Continuous Deployment) pipeline or workflow is a set of automated processes that are used to build, test, and deploy software. The pipeline is designed to ensure that code is tested and deployed quickly and efficiently, while minimizing the risk of errors and downtime.

In short, CI is a set of practices performed as developers are writing code, and CD is a set of practices performed after the code is completed.

A typical CICD workflow is composed of several stages, each with a specific purpose. The main stages of a CICD workflow are:

1. Build: This stage is used to compile the code, create executable files, and package the application.
2. Test: This stage is used to run automated tests to ensure that the application is working as expected. This may include unit tests, integration tests, and end-to-end tests.
3. Deploy: This stage is used to deploy the application to a staging or production environment.
4. Monitor: This stage is used to monitor the deployed application for errors and performance issues.

How does CI/CD relate to DevOps?

The CI/CD pipeline is part of the broader DevOps/DevSecOps framework. In order to successfully implement and run a CI/CD pipeline, organizations need tools to prevent points of friction that slow down integration and delivery. Teams require an integrated toolchain of technologies to facilitate collaborative and unimpeded development efforts.

GitHub Actions

GitHub Actions is a continuous integration and continuous delivery (CI/CD) platform that allows you to automate your build, test, and deployment pipeline. You can create workflows that build and test every pull request to your repository, or deploy merged pull requests to production.

GitHub Actions goes beyond just DevOps and lets you run workflows when other events happen in your repository. For example, you can run a workflow to automatically add the appropriate labels whenever someone creates a new issue in your repository.

GitHub provides Linux, Windows, and macOS virtual machines to run your workflows, or you can host your own self-hosted runners in your own data center or cloud infrastructure.

You only need a GitHub repository to create and run a GitHub Actions workflow. You need to add a workflow that demonstrates some of the essential features of GitHub Actions.

Lets try an example that will shows you how GitHub Actions jobs can be automatically triggered, where they run, and how they can interact with the code in your repository.

Activity

Create new repository

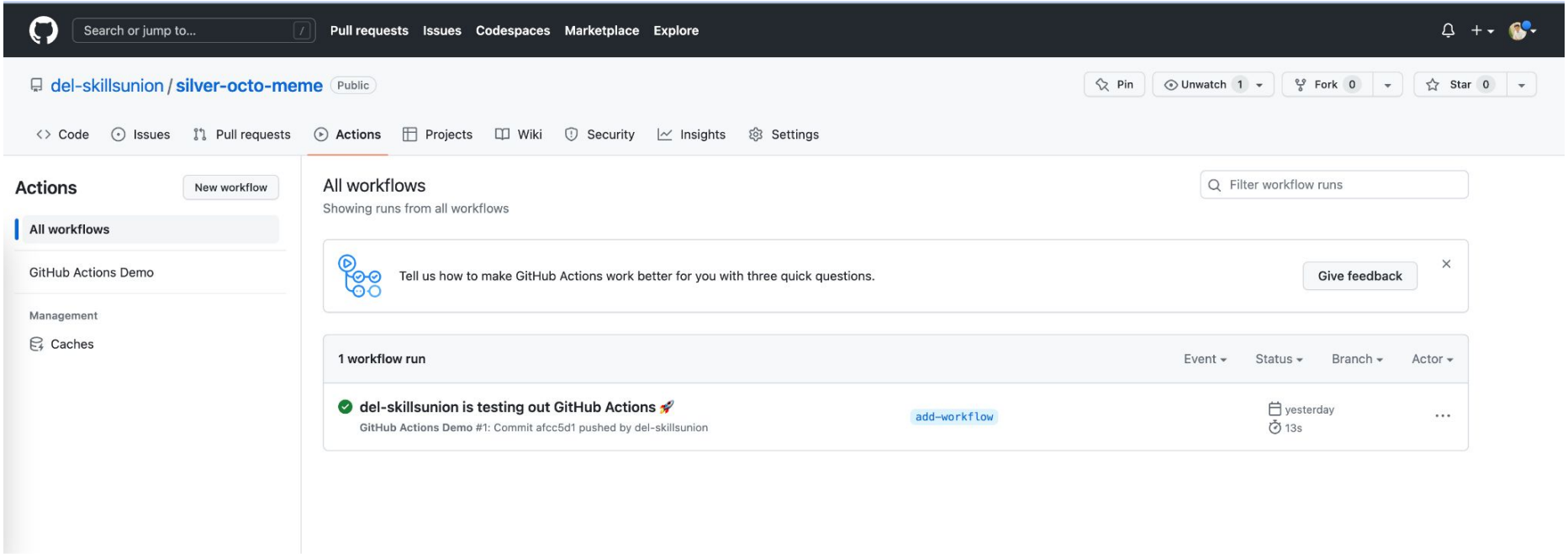
Learners spend 3-5 mins to create new public repository

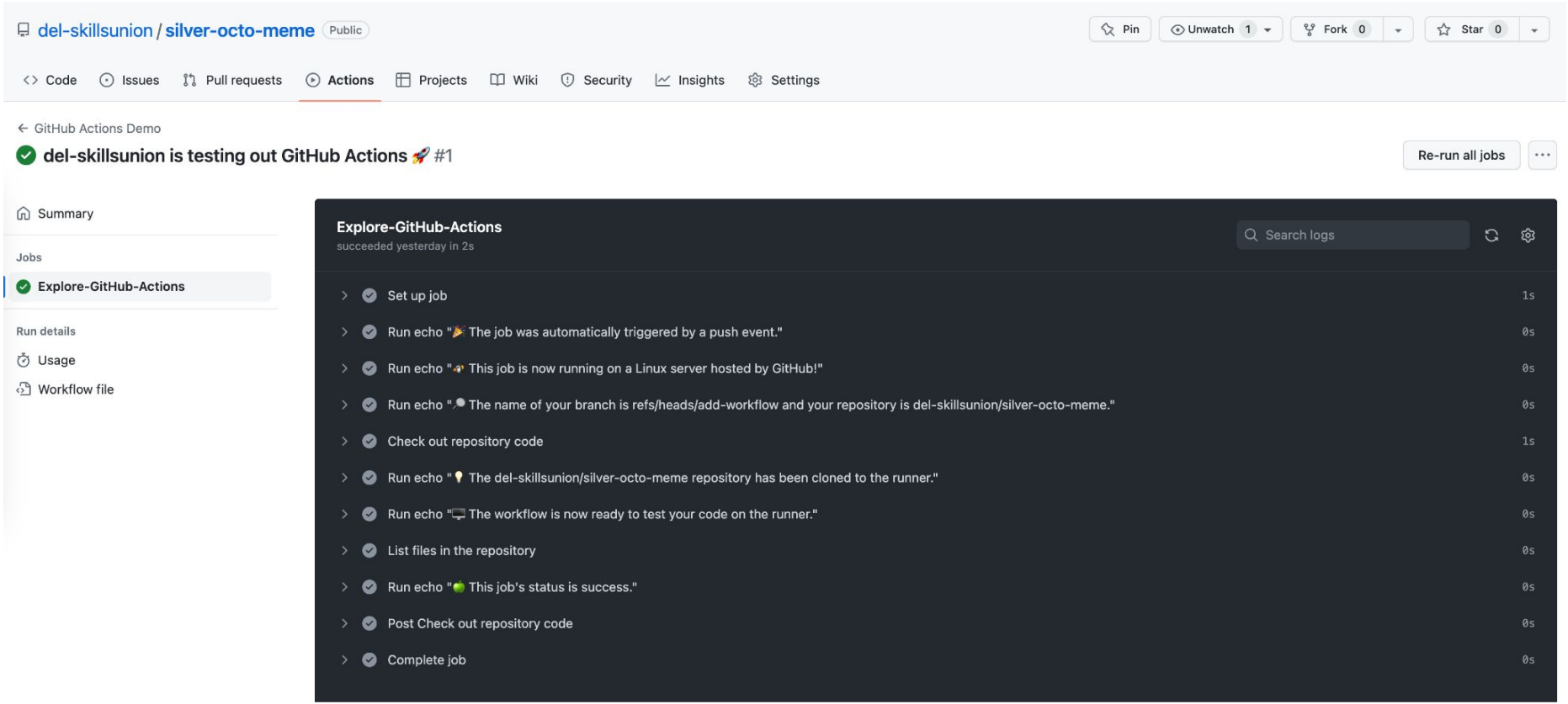
Activity

Instructor Demo GitHub Actions

- Create a .github/workflows directory in your repository on GitHub if this directory does not already exist.
- In the .github/workflows directory, create a file named github-actions-demo.yml.
- Copy the YAML contents into the github-actions-demo.yml (in sample app on the repository)
- Instructor demo where to see the GitHub actions progress.

Link: 6m-cloud-3.9-introduction-to-devops/github-actions-demo.yml at main · su-ntu-ctp/6m-cloud-3.9-introduction-to-devops


Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
del-skillsunion / silver-octo-meme
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Actions
New workflow
All workflows
GitHub Actions Demo
Management
Caches
All workflows
Showing runs from all workflows
Filter workflow runs
Tell us how to make GitHub Actions work better for you with three quick questions.
Give feedback
1 workflow run
Event
Status
Branch
Actor
del-skillsunion is testing out GitHub Actions 🚀
GitHub Actions Demo #1: Commit afcc5d1 pushed by del-skillsunion
add-workflow
yesterday
13s


del-skillsunion / silver-octo-meme
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
GitHub Actions Demo
del-skillsunion is testing out GitHub Actions 🚀 #1
Re-run all jobs
Summary
Jobs
Explore-GitHub-Actions
Run details
Usage
Workflow file
Explore-GitHub-Actions
succeeded yesterday in 2s
Search logs
Set up job 1s
Run echo "🎉 The job was automatically triggered by a push event." 0s
Run echo "🖥️ This job is now running on a Linux server hosted by GitHub!" 0s
Run echo "🔎 The name of your branch is refs/heads/add-workflow and your repository is del-skillsunion/silver-octo-meme." 0s
Check out repository code 1s
Run echo "💡 The del-skillsunion/silver-octo-meme repository has been cloned to the runner." 0s
Run echo "🖥️ The workflow is now ready to test your code on the runner." 0s
List files in the repository 0s
Run echo "🍏 This job's status is success." 0s
Post Check out repository code 0s
Complete job 0s

Activity

Learners try GitHub Actions for 15-20 mins

- Create a .github/workflows directory in your repository on GitHub if this directory does not already exist.
- In the .github/workflows directory, create a file named github-actions-demo.yml.
- Copy the YAML contents into the github-actions-demo.yml (in sample app on the repository)

The components of GitHub Actions

You can configure a GitHub Actions workflow to be triggered when an event occurs in your repository, such as a pull request being opened or an issue being created. Your workflow contains one or more jobs which can run in sequential order or in parallel. Each job will run inside its own virtual machine runner, or inside a container, and has one or more steps that either run a script that you define or run an action, which is a reusable extension that can simplify your workflow.

Component:

- Workflows
- Events
- Jobs
- Actions
- Runners

Workflows

A workflow is a configurable automated process that will run one or more jobs. Workflows are defined by a YAML file checked in to your repository and will run when triggered by an event in your repository, or they can be triggered manually, or at a defined schedule.

Workflows are defined in the .github/workflows directory in a repository, and a repository can have multiple workflows, each of which can perform a different set of tasks. For example, you can have one workflow to build and test pull requests, another workflow to deploy your application every time a release is created, and still another workflow that adds a label every time someone opens a new issue.

Events

An event is a specific activity in a repository that triggers a workflow run. For example, activity can originate from GitHub when someone creates a pull request, opens an issue, or pushes a commit to a repository. You can also trigger a workflow to run on a schedule, by posting to a REST API, or manually.

Jobs

A job is a set of steps in a workflow that is executed on the same runner. Each step is either a shell script that will be executed, or an action that will be run. Steps are executed in order and are dependent on each other. Since each step is executed on the same runner, you can share data from one step to another. For example, you can have a step that builds your application followed by a step that tests the application that was built.

Actions

An action is a custom application for the GitHub Actions platform that performs a complex but frequently repeated task. Use an action to help reduce the amount of repetitive code that you write in your workflow files. An action can pull your git repository from GitHub, set up the correct toolchain for your build environment, or set up the authentication to your cloud provider.

You can write your own actions, or you can find actions to use in your workflows in the GitHub Marketplace.

Runners

A runner is a server that runs your workflows when they're triggered. Each runner can run a single job at a time. GitHub provides Ubuntu Linux, Microsoft Windows, and macOS runners to run your workflows; each workflow run executes in a fresh, newly-provisioned virtual machine. GitHub also offers larger runners, which are available in larger configurations. For more information, see "Using larger runners." If you need a different operating system or require a specific hardware configuration, you can host your own runners.

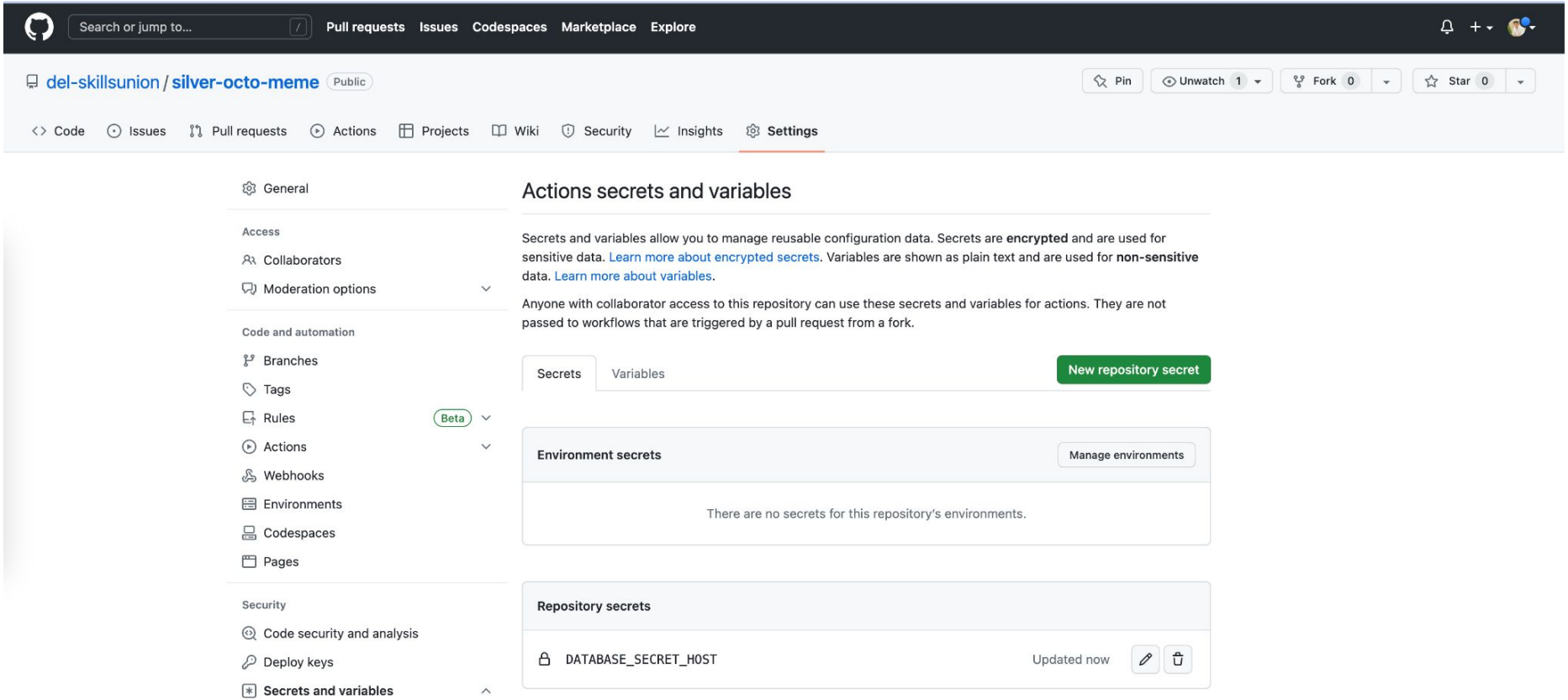
GitHub Secrets Configuration

GitHub Secrets Configuration

Navigate to the “Settings” section in our GitHub repository and locate the “Secrets” section on the left-hand side. By clicking the “new repository secret” we are going to add these two secrets.

Activity:

- Create any secret on GitHub Secret, then run echo on the workflow


Search or jump to...
Pull requests
Issues
Codespaces
Marketplace
Explore
del-skillsunion / silver-octo-meme
Public
Pin
Unwatch 1
Fork 0
Star 0
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Beta
Actions
Webhooks
Environments
Codespaces
Pages
Security
Code security and analysis
Deploy keys
Secrets and variables
Actions secrets and variables
Secrets and variables allow you to manage reusable configuration data. Secrets are encrypted and are used for sensitive data. Learn more about encrypted secrets. Variables are shown as plain text and are used for non-sensitive data. Learn more about variables.
Anyone with collaborator access to this repository can use these secrets and variables for actions. They are not passed to workflows that are triggered by a pull request from a fork.
Secrets
Variables
New repository secret
Environment secrets
Manage environments
There are no secrets for this repository's environments.
Repository secrets
DATABASE_SECRET_HOST
Updated now

GitHub Secrets Configuration

```
name: GitHub Actions Secrets

run-name: ${{ github.actor }} is testing out GitHub Action🚀

on: [push]

jobs:

  Explore-GitHub-Secrets:

    runs-on: ubuntu-latest

    steps:

      - name: Printing out secret

        env:

          MY_SECRET: ${{ secrets.DATABASE_SECRET_HOST}}

        run: echo "My secret is ${MY_SECRET}"
```

Link: [6m-cloud-3.9-introduction-to-devops/secrets.yml at main · su-ntu-ctp/6m-cloud-3.9-introduction-to-devops (github.com)]

Activity

Learner:

- Clean up AWS.
- Remove/delete/terminate all service/ resources that you created.

Instructor

- Clean up AWS.
- Remove/delete/terminate all service/ resources that you created.
- Check the AWS account after learner clean up.

GitHub Secrets Configuration

```
name: GitHub Actions Secrets

run-name: ${{ github.actor }} is testing out GitHub Action🚀

on: [push]

jobs:

  Explore-GitHub-Secrets:

    runs-on: ubuntu-latest

    steps:

      - run: echo "🎉 The secret is pulled from GitHub Secret ${{ secrets.DATABASE_SECRET_HOST }}"

      - run: echo "🍏 This job's status is ${{ job.status }}."
```

Q1: According to Agile manifesto -

A: Individuals and interactions over people and technique
B: Individuals and interactions over projects and tools
C: Individuals and interactions over processes and tools.
D: Individuals and interactions over products and tools

AWS DevOps

AWS DevOps Tools

AWS (Amazon Web Services) provides a wide range of tools for implementing DevOps in the cloud, including:

- AWS Cloud Development Kit (CDK): an open-source framework for modeling and provisioning cloud resources using familiar programming languages.
- AWS CodeBuild: a service for building and testing code with continuous scaling.
- AWS CodeDeploy: a tool for automating software deployments to various services such as Amazon EC2, AWS Fargate, AWS Lambda, or on-premises servers.
- AWS CodePipeline: a tool for automating the continuous delivery of code for rapid and accurate updates.
- AWS CodeStar: a comprehensive DevOps tool for developing, building, and deploying applications on AWS, with an easy-to-use interface.
- AWS Device Farm: a tool for testing web and mobile apps across real mobile devices and desktop browsers hosted in the AWS cloud, allowing developers to improve the quality of their apps.