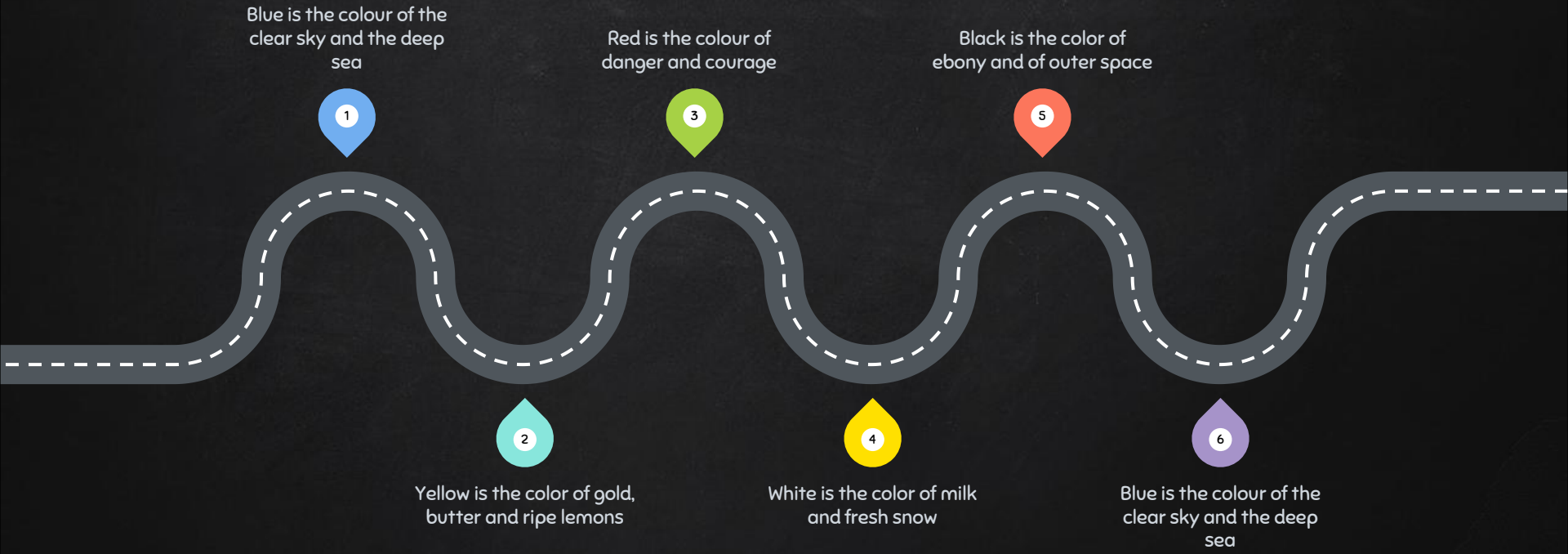
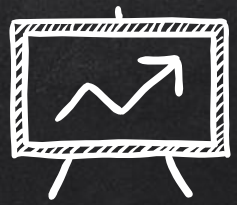
Thank You

Get Your Inspiration Within

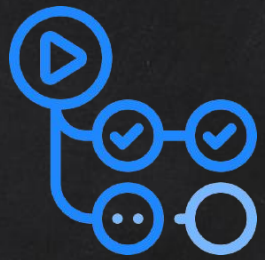
THE COFFEE BREAK

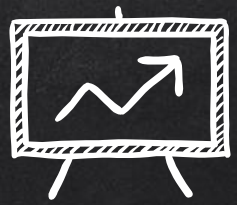
Roadmap


Blue is the colour of the clear sky and the deep sea
1
Yellow is the color of gold, butter and ripe lemons
2
Red is the colour of danger and courage
3
White is the color of milk and fresh snow
4
Black is the color of ebony and of outer space
5
Blue is the colour of the clear sky and the deep sea
6



Learning GitHub Actions





Git Overview

Git

- ✘ Git is a distributed version control system (DVCS) that allows multiple developers or other contributors to work on a project.
- ✘ It provides a way to work with one or more local branches and push them to a remote repository.
- ✘ Git is responsible for everything GitHub-related that happens locally on your computer. Key features provided by Git include:

Common Git Commands

git init

to start a new empty repository or to reinitialize an existing one in the project root.

git clone

used for downloading the latest version of a remote project to the local machine

git checkout

to switch the branch that you are currently working on

git commit

To save changes or take a snapshot of the current state of the branch,

git pull

changes from the remote repository and merge any remote changes in the current local branch

git push

to upload local repository content to a remote repository

git status

git add .

git commit –m "commit message"

git pull

git push

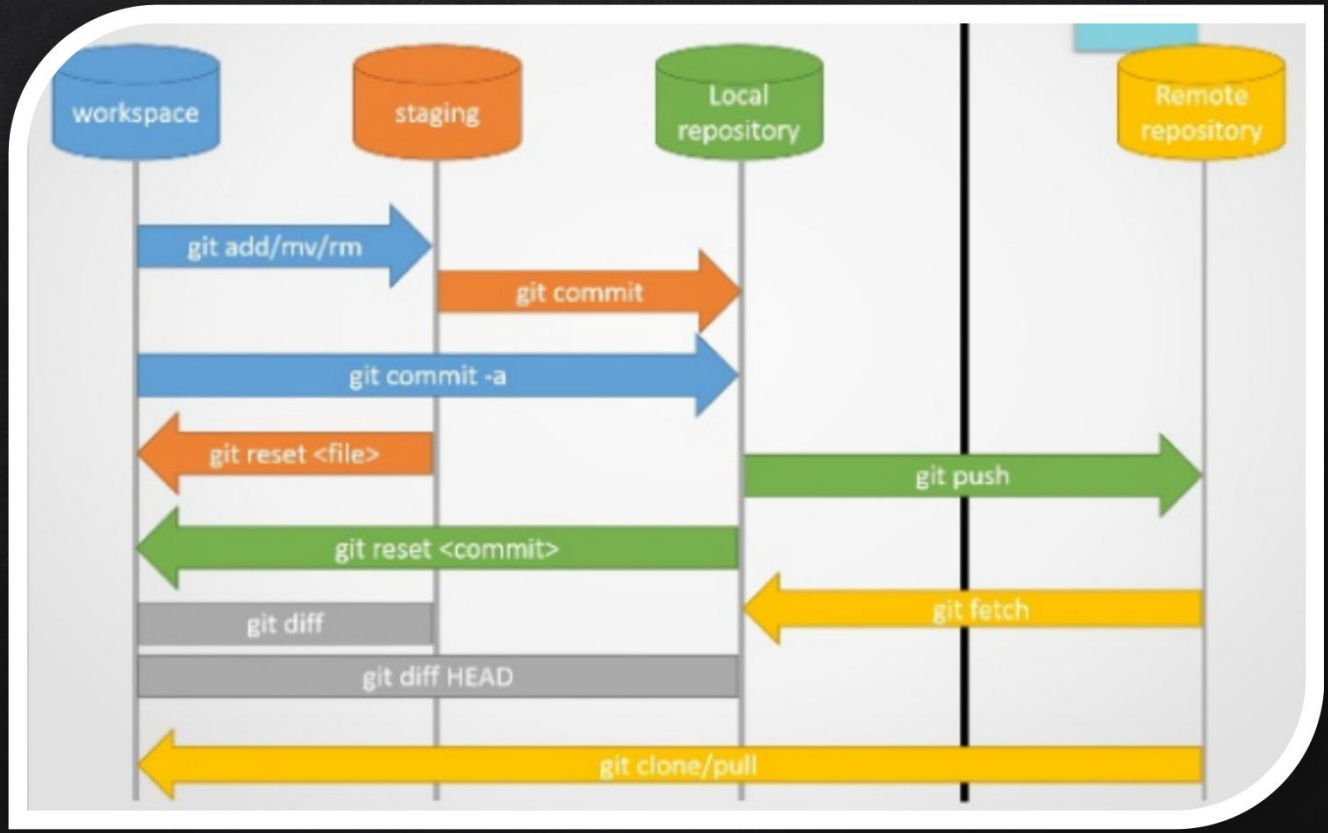
git log

git branch

git checkout

git merge

Git Architecture



GitHub

GitHub is a cloud platform that uses Git as its core technology. It simplifies the process of collaborating on projects and provides a website, command-line tools, and overall flow that allows developers and users to work together. GitHub acts as the "remote repository" in the Git section.

Key features provided by GitHub include

- ✘ Issues
- ✘ Discussions
- ✘ Pull requests
- ✘ Notifications
- ✘ Labels
- ✘ Actions
- ✘ Forks
- ✘ Projects

Key features provided by GitHub include

Issues

Is the color of gold, butter and ripe lemons. In the spectrum of visible light, yellow is found between green and orange.

Blue

Is the colour of the clear sky and the deep sea. It is located between violet and green on the optical spectrum.

Red

Is the color of blood, and because of this it has historically been associated with sacrifice, danger and courage.

Yellow

Is the color of gold, butter and ripe lemons. In the spectrum of visible light, yellow is found between green and orange.

Blue

Is the colour of the clear sky and the deep sea. It is located between violet and green on the optical spectrum.

Red

Is the color of blood, and because of this it has historically been associated with sacrifice, danger and courage.

GitHub

GitHub is a development platform that enables you to host and review code, manage projects, and build software alongside 50 million developers.

Issues

Issues are where most of the communication between a project's consumers and development team occurs.

An issue can be created to discuss a broad set of topics, including bug reports, feature requests, documentation clarifications, and more.

Once an issue has been created, it can be assigned to owners, labels, projects, and milestones. You can also associate issues with pull requests and other GitHub items to provide future traceability.

Commits

- A commit is a change to one or more files on a branch. Every time a commit is created, it's assigned a unique ID and tracked, along with the time and contributor.

- This provides a clear audit trail for anyone reviewing the history of a file or linked item, such as an issue or pull request.

Discussions

GitHub Discussions is a collaborative communication forum for the community around an open source project. Community members can ask and answer questions, share updates, have open-ended conversations, and follow along on decisions affecting the community's way of working.

Pull Requests

A pull request is the mechanism used to signal that the commits from one branch are ready to be merged into another branch.

GitHub Tags

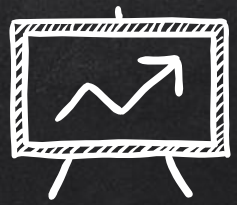
Tags are associated with commits, so you can use a tag to mark an individual point in your repository's history, including a version number for a release.

Labels

Labels provide a way to categorize and organize issues and pull requests in a repository.

As you create a GitHub repository several labels will automatically be added for you and new ones can also be created.

Examples: bug, documentation, duplicate, help wanted, enhancement, question



Introduction to Workflows

Understanding 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.

What is GitHub Action Workflow

- 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
- With a workflow, you can build, test, package, release, and deploy any project on GitHub.

! A Simple Workflow.yml U ×

.github > workflows > ! A Simple Workflow.yml

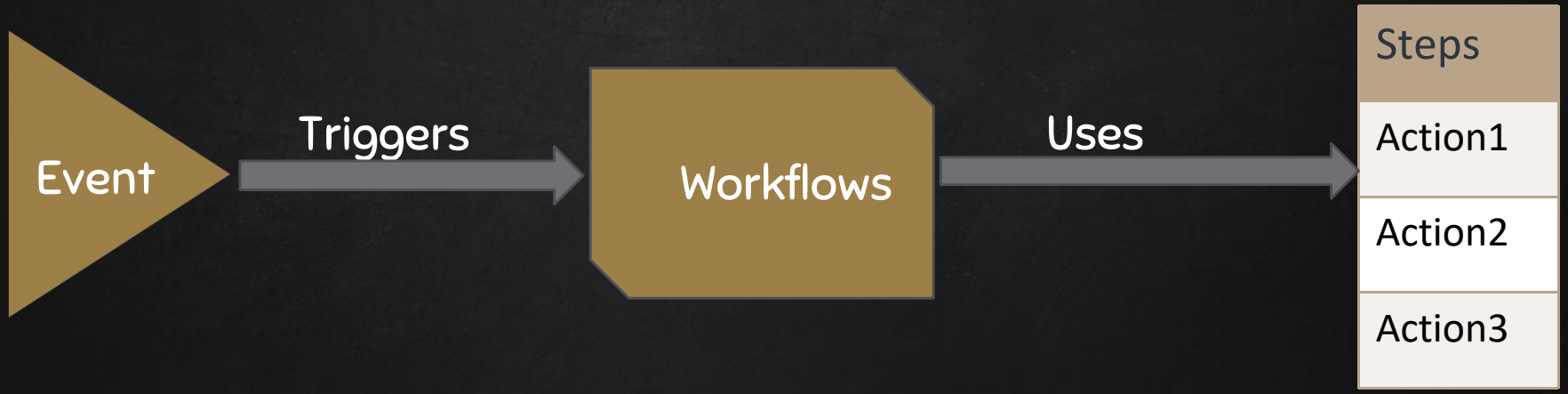
```yaml
# This is a simple workflow

name: 'GitHub Workflow Demo'

on: workflow_dispatch

jobs:
  sayHello:
    runs-on: ubuntu-latest
    steps:
    - name: runscript-to-say-hello
      run:  echo "Howdy GitHub Actions"
```

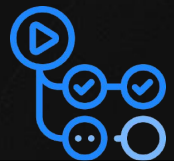
Common DevOps Tasks

- ✘ Add new contributors
- ✘ Close stale issues & PRs
- ✘ Pull request
- ✘ Organizational tasks: Issue created by users
- ✘ Label the issue { minor, major, reproducible, priority etc. }
- ✘ Fix the issue and raise a PR
- ✘ Verify PR and merge the code to master
- ✘ New release with a version number
- ✘ Build pipeline to test, and build code

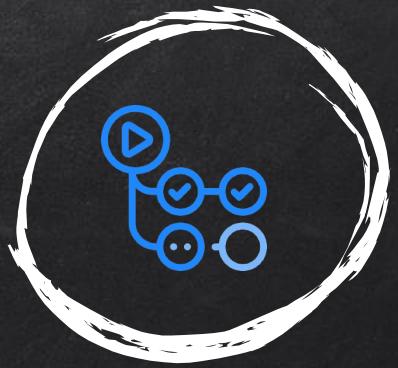
GitHub Action Workflow

Work + Flow





- ✓ GitHub Action Workflow Directory: .github/workflows/
- ✓ File Extension: .yml



Components of GitHub Action Workflow

Vijay Saini

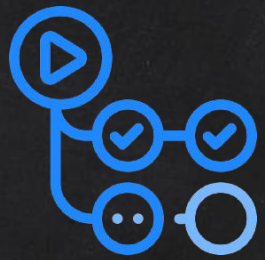
Events

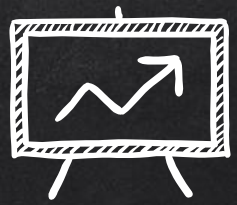
- An event is a specific activity in a repository that triggers a workflow run.
- For example, activity can originate from GitHub when someone creates a pull request, opens an issue, or pushes a commit to a repository.
- You can also trigger a workflow run on a schedule, by posting to a REST API, or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

```yaml
on:
  # Triggers the workflow on push or pull request events but only for the "dev" branch
  push:
    branches: [ "dev" ]
  pull_request:
    branches: [ "dev" ]

  # Allows you to run this workflow manually from the Actions tab
  workflow_dispatch:
```

```yaml
name: 'GitHub Workflow Demo'

# 'on' is used to define the trigger of this workflow
on:
  # Below 'push' trigger is to ensure this workflow runs when a  new push happens
  # in a branch named "dev" and file at path .github/workflows/ is modified.
  push:
    branches:
    - "dev"
    paths:
    - ".github/workflows/*"

  # Below 'pull_request' trigger is to ensure this workflow runs on a pull_request event
  # for any branch except those whose name starts with test/exp*
  pull_request:
    branches-ignore:
    - 'test/exp*'

  # workflow-dispatch is to ensure we can run this workflow manually
  workflow_dispatch:
    environment:
      description: 'Environment to run tests against'
      type: environment
      required: true
```

```yaml
on:
  workflow_dispatch:
    inputs:
      logLevel:
        description: 'Log level'
        required: true
        default: 'warning'
        type: choice
        options:
        - info
        - warning
        - debug
      print_tags:
        description: 'True to print to STDOUT'
        required: true
        type: boolean
      tags:
        description: 'Test scenario tags'
        required: true
        type: string
      environment:
        description: 'Environment to run tests against'
        type: environment
        required: true
```





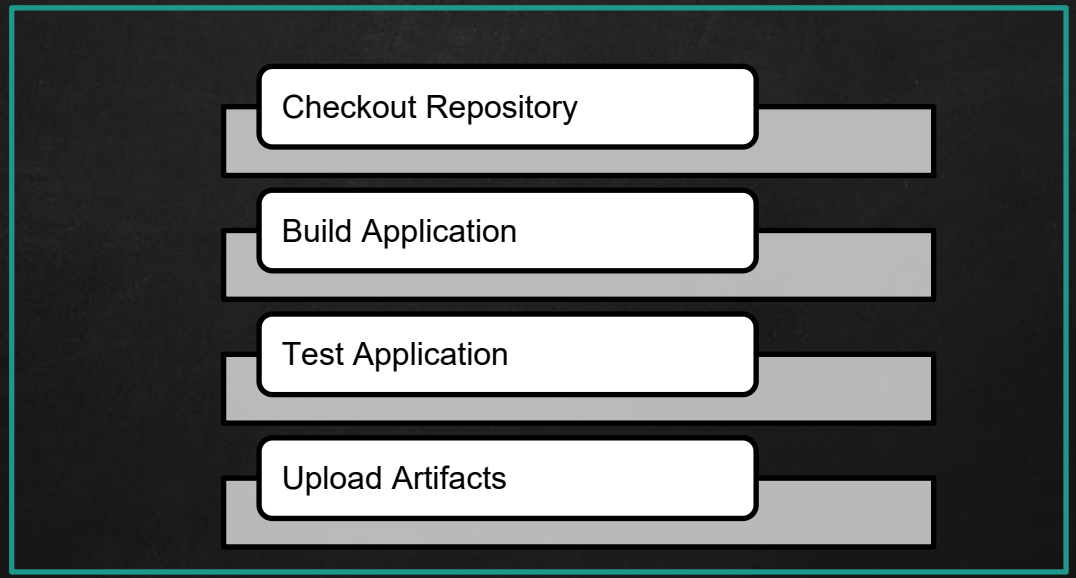
GitHub Jobs

GitHub Jobs

A job is a set of steps in a workflow that execute on the same runner. Each step is either a shell script that will be executed, or an action that will be run.

Step

Steps can run commands, run setup tasks, or run an action



What is GitHub Action

- 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.

What is GitHub Action

- ✘ GitHub Actions helps you automate your software development workflows from within GitHub. You can deploy workflows in the same place where you store code and collaborate on pull requests and issues.

- ✘ GitHub Actions optimize code delivery time from idea to deployment on a community-powered platform.

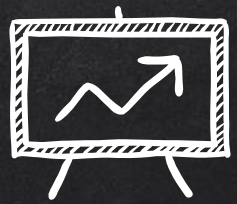
GitHub Workflow Summary

- ✘ Workflow is an automated process that you set up in your GitHub repository.
- ✘ A workflow needs to have at least one job and can be triggered by different events.
- ✘ You can build, test, package, release, or deploy any project on GitHub with a workflow.

```yaml
# This is a simple workflow

name: 'GitHub Workflow Demo'

on: workflow_dispatch

jobs:
  sayHello:
    runs-on: ubuntu-latest
    steps:
    - name: runscript-to-say-hello
      run:  echo "Howdy GitHub Actions"

```

Lessons Learned

- ✘ How to create a workflow
- ✘ Event Trigger
- ✘ Job
- ✘ Step
- ✘ Runner
- ✘ How to run a workflow
- ✘ View Workflow run logs



Git Action

GitHub Action

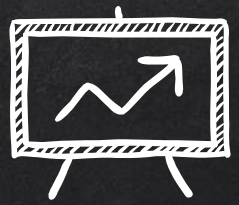
An action is a custom application for the GitHub Actions platform that performs a complex but frequently repeated task.

Use an action to help reduce the amount of repetitive code that you write in your workflow files.

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

GitHub Action

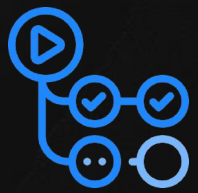
dd



GitHub Runner

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.



Runner Types

GitHub Hosted Runner

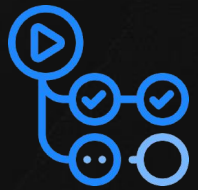
- ✘ GitHub automatically provisions a new VM with Ubuntu Linux, Windows, or macOS operating systems
- ✘ Machine maintenance and upgrades are taken care by GitHub
- ✘ Free GitHub plan + per-minute rates
- ✘ Provide a clean instance for every job execution.

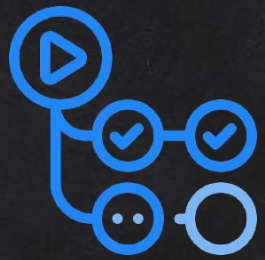
Self Hosted Runner

- ✘ A self-hosted runner is a system that you deploy and manage to execute jobs from GitHub Actions
- ✘ Customizable to your hardware, operating system, software, and security requirements
- ✘ Are free to use with GitHub Actions
- ✘ Don't need to have a clean instance for every job execution.

Why Self-hosted Runners

- ✘ Software Dependency
- ✘ VM Size and Configuration
- ✘ Secure Access & Networking
- ✘ Large Workload Support





Programming Variable

Variables are used to store information to be referenced and manipulated in a program.

ApplicationService = 'http://github-action-is-awesome.com'

Create http://github-action-is-awesome.com

Deploy my app to http://github-action-is-awesome.com

Perform UAT on http://github-action-is-awesome.com

Perform Load testing on http://github-action-is-awesome.com

- Create ApplicationService
- Deploy my app to ApplicationService
- Perform UAT on ApplicationService
- Perform Load testing on ApplicationService

Environment Variables

We can use environment variables to store information that you want to reference in your workflow.

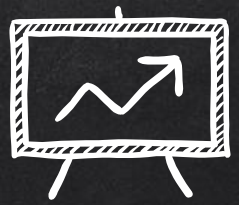
```yaml
env:
  DAY_OF_WEEK: Funday
  testvar: value-1
```

```yaml
run: |
  echo "$Greeting $First_Name. Today is $DAY_OF_WEEK!"
  echo "testvar  $testvar"
  echo "testvar  ${{ env.testvar }}"
```

GitHub Secretes

Secrets are encrypted environment variables that you create in an organization, repository, or repository environment. The secrets that you create are available to use in GitHub Actions workflows.

GitHub uses a libsodium sealed box to help ensure that secrets are encrypted before they reach GitHub and remain encrypted until you use them in a workflow.



GitHub Environments

Using environments for deployment

We can configure environments with protection rules and secrets.

A workflow job that references an environment must follow any protection rules for the environment before running or accessing the environment's secrets.

GitHub Secretes

APP_SECRET

PRODUCTION_APP_SECRET

STAGE_APP_SECRET

QA_APP_SECRET

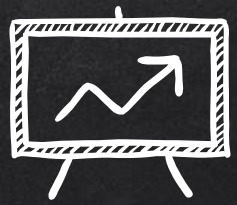
UAT_APP_SECRET

DEV_APP_SECRET

GitHub Environments

Environments are used to describe a general deployment target like production, staging, or development.

You can configure environments with protection rules and secrets. A workflow job that references an environment must follow any protection rules for the environment before running or accessing the environment's secrets.



Artifacts

Artifacts

Artifacts allow us to share data between jobs in a workflow and store data once that workflow has completed.

These are some of the common artifacts that we can upload:

- Log files and core dumps
- Test results, failures, and screenshots
- Binary or compressed files
- Stress test performance output and code coverage results

Artifact storage

Artifacts are stored in storage space on GitHub. The space is free for public repositories and some amount is free for private repositories, depending on the account. GitHub stores your artifacts for 90 days.

In the following workflow snippet, notice that in the `actions/upload-artifact@main` action there is a `path:` attribute. This is the path to store the artifact. Here, we specify *public/* to upload everything to a directory. If it was just a file that we wanted to upload, we could use something like *public/mytext.txt*.

yml Copy

```yml
build:
  runs-on: ubuntu-latest
  steps:
    - uses: actions/checkout@v1
    - name: npm install and build webpack
      run: |
        npm install
        npm run build
    - uses: actions/upload-artifact@main
      with:
        name: webpack artifacts
        path: public/
```

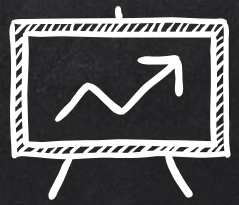
Artifacts

Uploading files: Give the uploaded file a name and upload the data before the job ends.

Downloading files: You can only download artifacts that were uploaded during the same workflow run. When you download a file, you can reference it by name.

Artifacts

```yaml
name: Share data between jobs
on: push
jobs:
  job_1:
    name: Upload File
    runs-on: ubuntu-latest
    steps:
      - run: echo "Hello World" > file.txt
      - uses: actions/upload-artifact@v2
        with:
          name: file
          path: file.txt

  job_2:
    name: Download File
    runs-on: ubuntu-latest
    needs: job_1
    steps:
      - uses: actions/download-artifact@v2
        with:
          name: file
      - run: cat file.txt

```



Infra Automation

Infrastructure Requirement

	Resource	Specification
1	Storage Account	SKU: Standard_LRS, kind: StorageV2
2	Container Registry	SKU: Basic, adminUserEnabled: true
3	Service Bus Namespace	Default (Basic Tier)
4	App Service Plan	SKU: Basic, Kind: Linux
5	Web App	Runtime: .Net 7

Infrastructure as a code

- ✘ Infrastructure as code (IaC) uses DevOps methodology and versioning with a descriptive model to define and deploy infrastructure
- ✘ define all infrastructure as version-controlled text files
- ✘ automate deployment and provisioning

Challenges in Manual Infra Deployment

- ✘ Inconsistencies in Deployments
- ✘ Repetitive & Complex Process
- ✘ Labor Intensive
- ✘ Human Error
- ✘ Slow Process

Why Infrastructure as a code ?

- ✘ Avoid manual configuration to enforce consistency
- ✘ Idempotence
- ✘ IaC boosts productivity through automation
- ✘ Faster speed and consistency

Local Envt Setup for Bicep development and deployment

VS Code	+	Bicep Extension
AZ CLI	+	Bicep Module

https://docs.microsoft.com/en-us/azure/azure-resource-manager/bicep/install

Bicep Decompile vs Build

```
# Convert ARM to Bicep

az bicep decompile --file azuredeploy.json


# Convert Bicep to ARM

az bicep build --file .\keyvault.bicep
```

Infra Deployment Automation

IaC Template

Setup Authentication

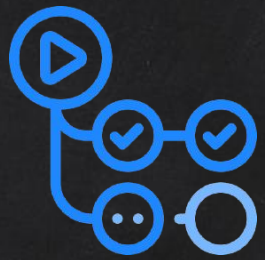
GitHub Action Workflow

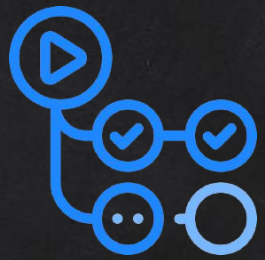
Azure AD service principals for automation authentication

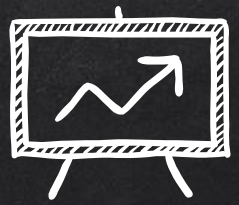
Create a service principal and configure its access to resource group:

```
az ad sp create-for-rbac --name github-devops-app --role contributor --scopes
/subscriptions/<subscription-d>/resourceGroups/<rg name> --sdk-auth
```

Note : It is always a good practice to grant minimum access.





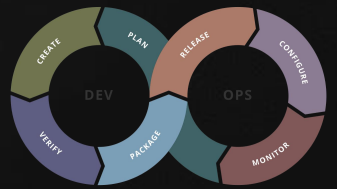


Continuous Integration

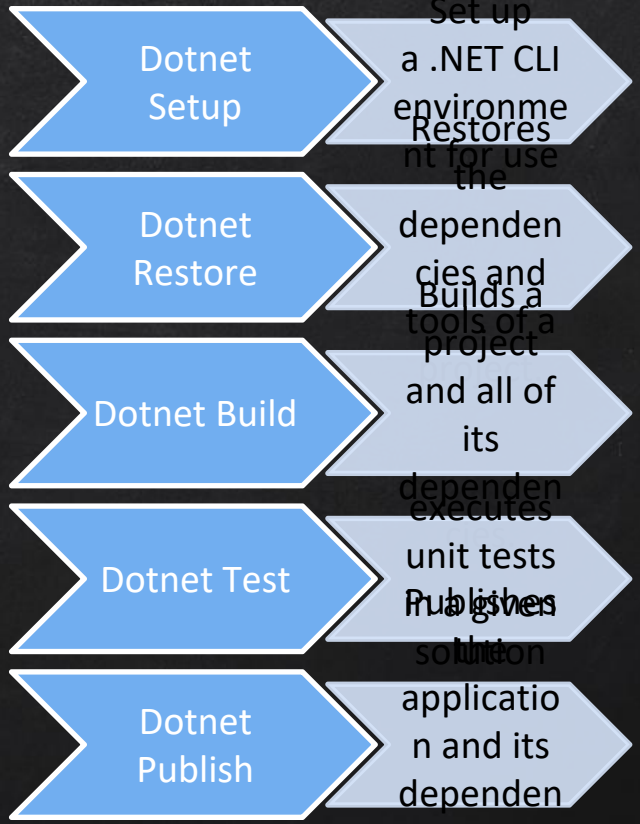
What is Continuous Integration

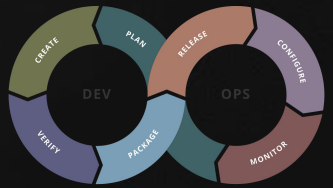
Continuous integration is a DevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run.

Continuous integration most often refers to the build or integration stage of the software release process.



CI : DOTNET PROJECT EXAMPLE



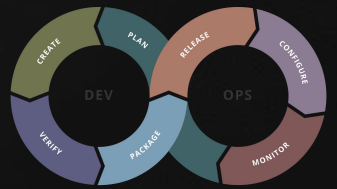


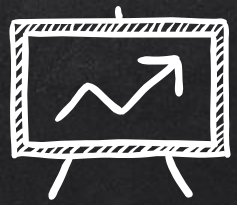
Git Ignore

.gitignore file is **a text file that tells Git which files or folders to ignore in a project.**

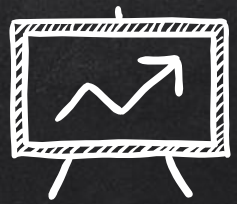
A local . gitignore file is usually placed in the root directory of a project. You can also create a global . gitignore file and any entries in that file will be ignored in all of your Git repositories.

dotnet new gitignore

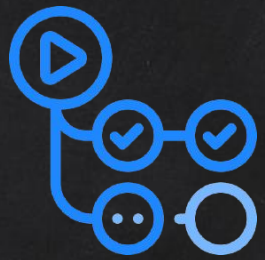




Continuous Deployment

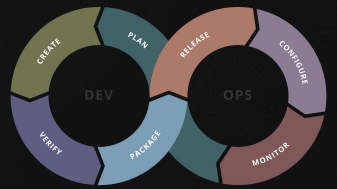


CICD Workflow for Deploying Docker Container

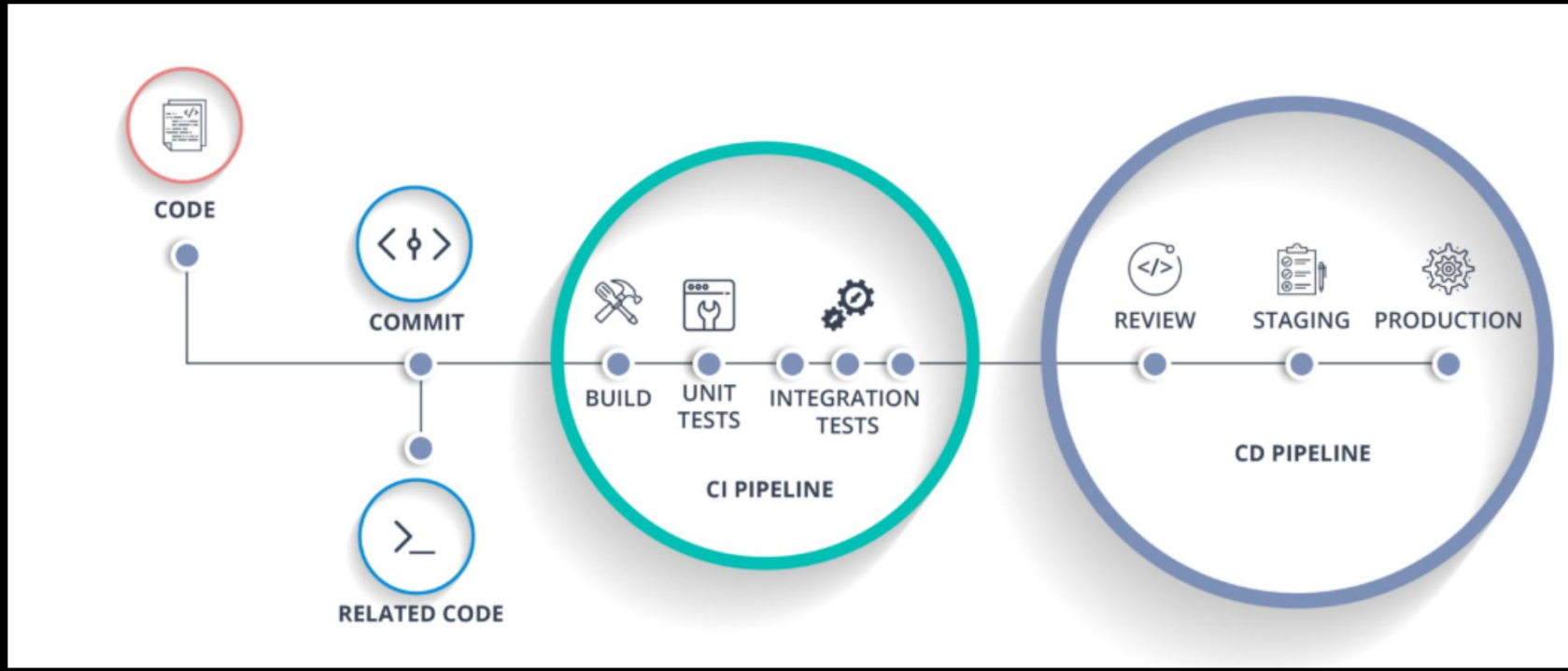


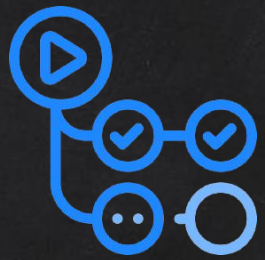
What is Continuous Deployment

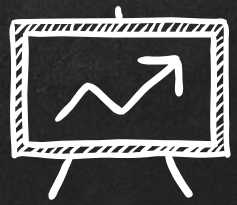
Continuous deployment (CD) is the practice of using automation to publish and deploy software updates. As part of the typical CD process, the code is automatically built and tested before deployment.



CI CD Workflow





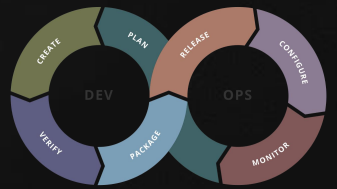


CICD Workflow for Deploying Docker Container

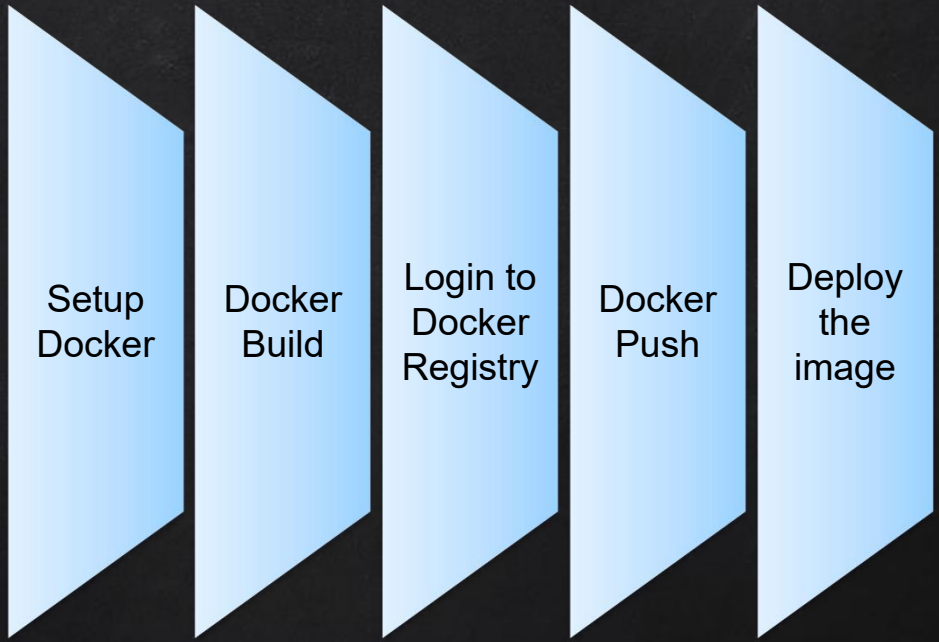
Requirement

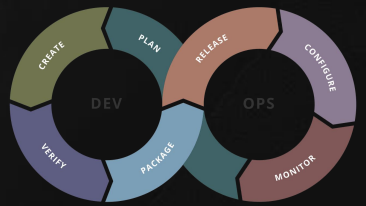
Your client TechSckool Ltd is building a microservices-based platform and wants you to design a GitHub Action Workflow to Build and deploy a Dotnet application's container to an Azure Web App.

You must use Azure Container Registry to manage container.



GitHub Action for Docker Deployment





custom actions

Vijay Saini

custom actions

Actions are individual tasks that you can combine to create jobs and customize your workflow. You can create your own actions, or use and customize actions shared by the GitHub community.

Composite Actions

A composite action allows you to combine multiple workflow steps within one action. For example, you can use this feature to bundle together multiple run commands into an action, and then have a workflow that executes the bundled commands as a single step using that action. To see an example, check out "Creating a composite action".