Core Tools

Git

Jenkins & GitHub

Actions

Docker

GIT

A Quick introduction to Version Control Systems and GIT

What Are Version Control systems

- A system that keeps records of your changes
-
- Allows for collaborative development
- Allows you to know who made what changes and when
- Allows you to revert any changes and go back to a previous state

Git icon

What Are Version Control systems

- Git is not the only one ☺

Git icon

Popularity

Features

Workflows

Distributed Nature

GIT Key Concepts

- SNAPSHOTS

The way git keeps track of your code history.

Essentially records what all your files look like at a given point in time

- COMMITS

Way of creating a snapshot.

A project is a set of commits

A Commit has 3 parts

1. A hash Code name
2. Information on what changed
3. A reference to the previous commit

GIT Key Concepts

- REPOSITORY

  The place where commits reside.
  Reside both on server and your machine ☺

- CLONING

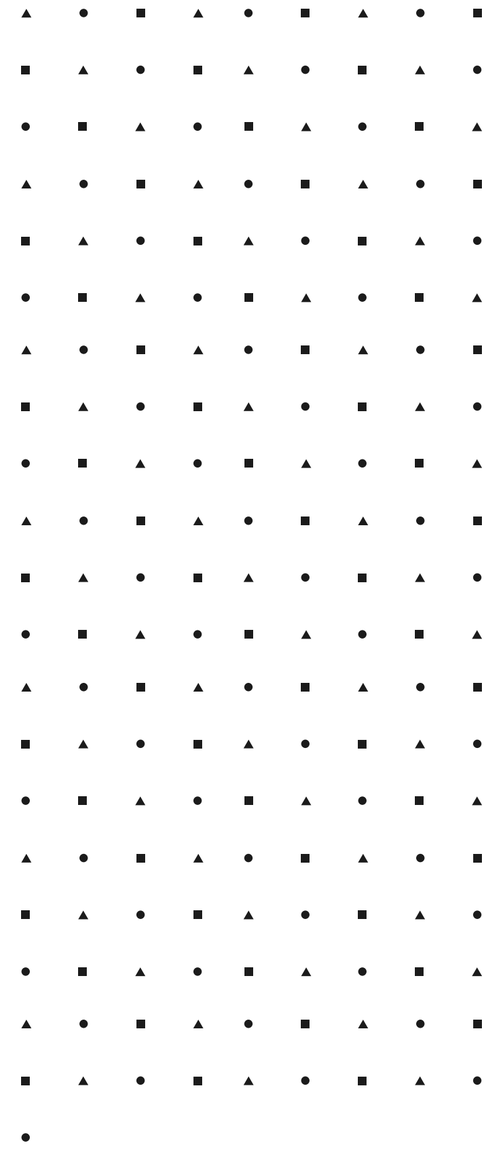
  The act to copying a repository from server to local
  See: git clone vs git init

- PULLING

  The act of getting changes from server to local
  See: git pull

- PUSHING

  The act of sending your changes from local to server
  See: git push



GIT Key Concepts

- BRANCHES

All commits in a git repository live in a branch.

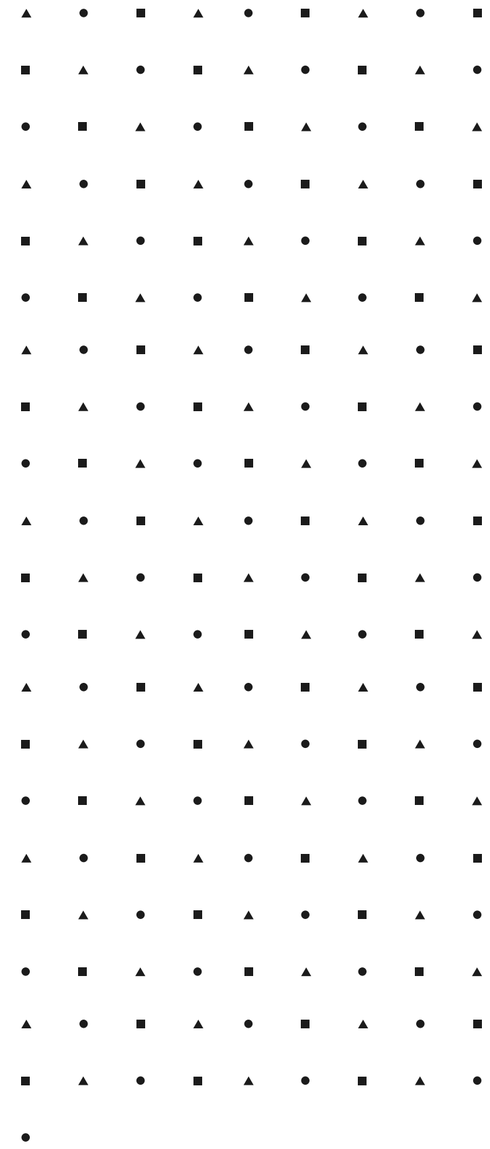
A Project can have multiple branches.

Usually main branch is called master or main. Just a convention

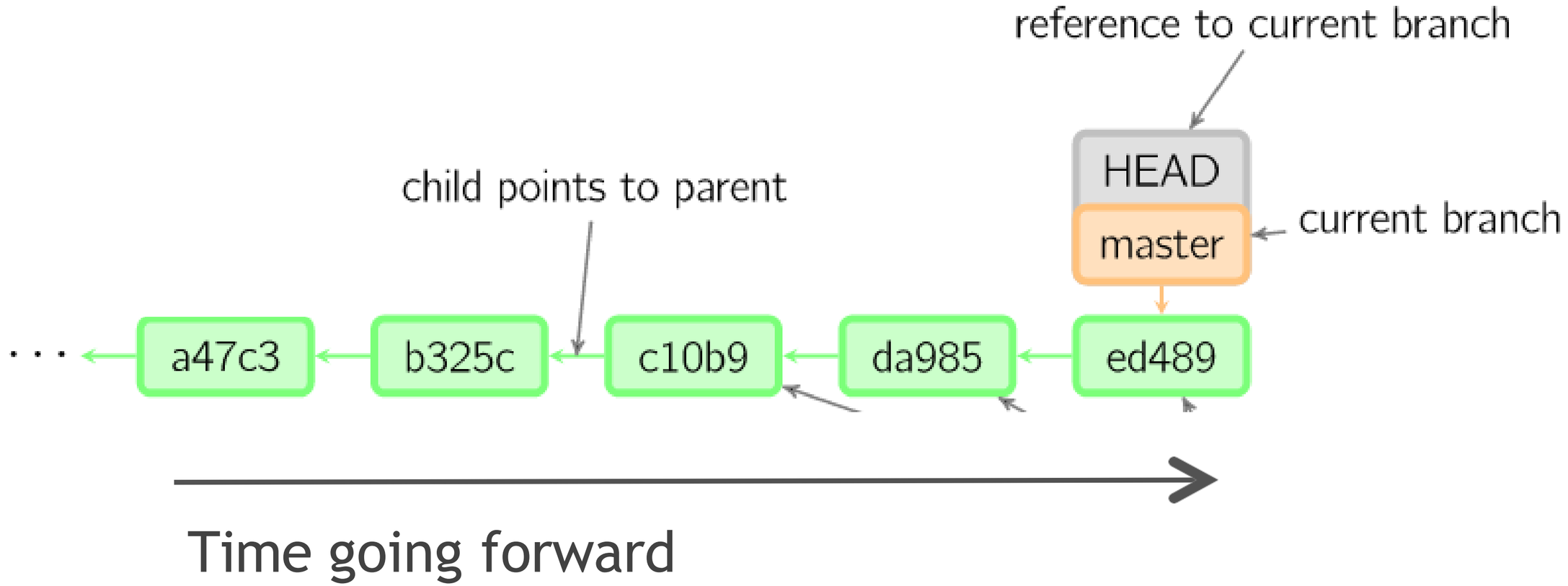
- MERGING

The act of combining changes in one branch with another.
See; git branch vs git rebase

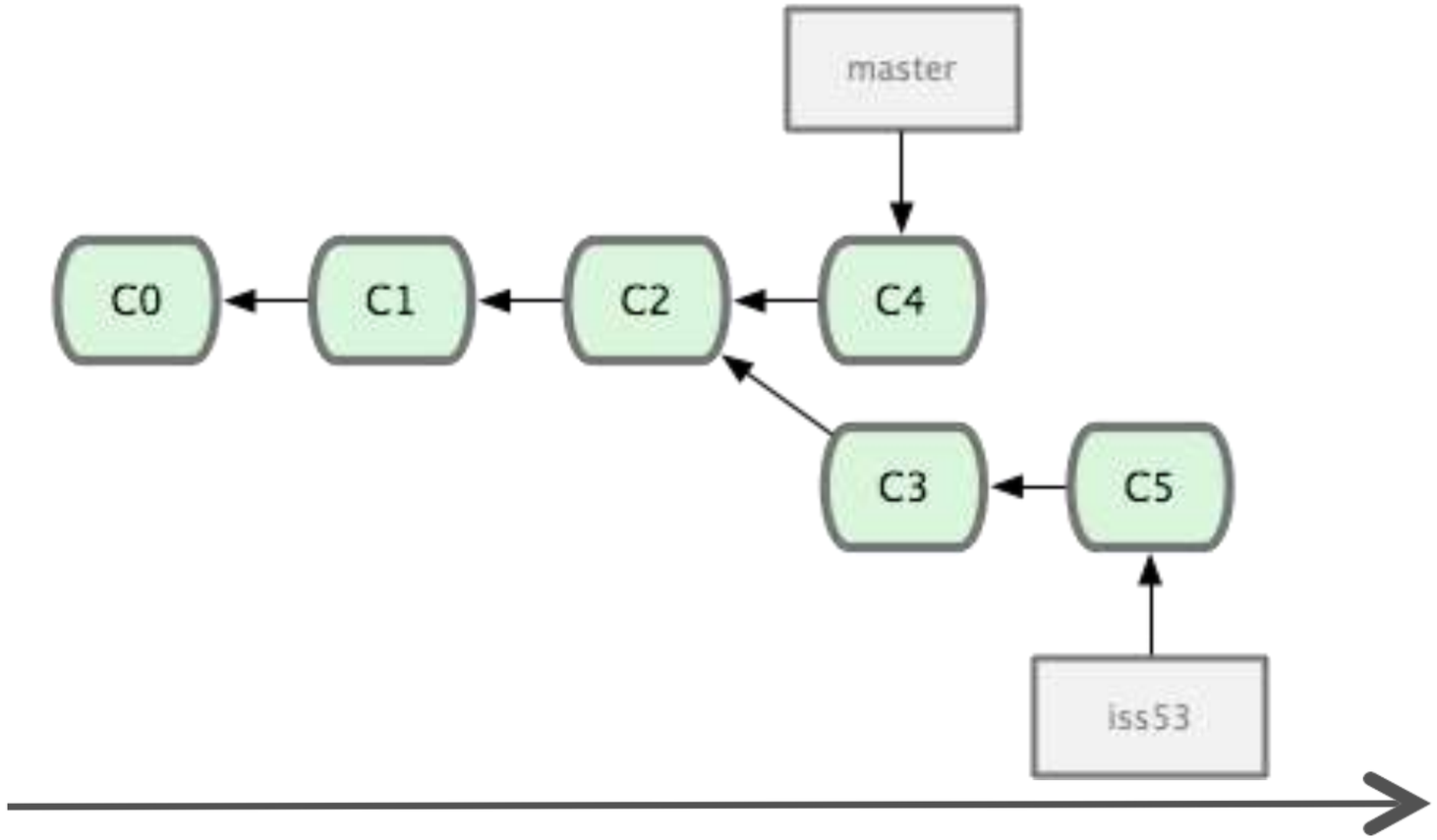
This act may cause conflicts



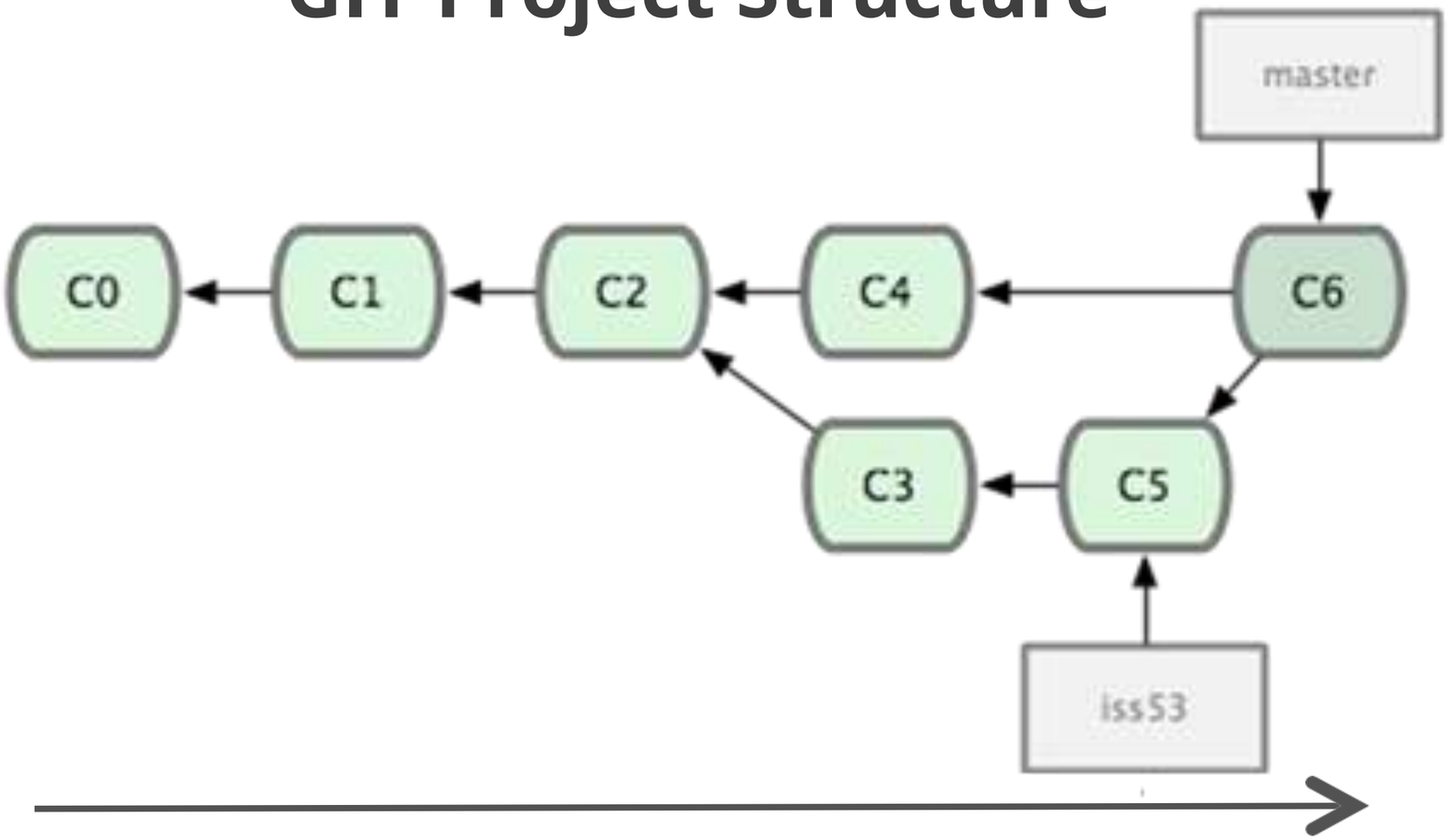
GIT Project Structure



GIT Project Structure



GIT Project Structure



GIT Key Concepts

- WORKING DIRECTORY

  The local directory where all your work is done.

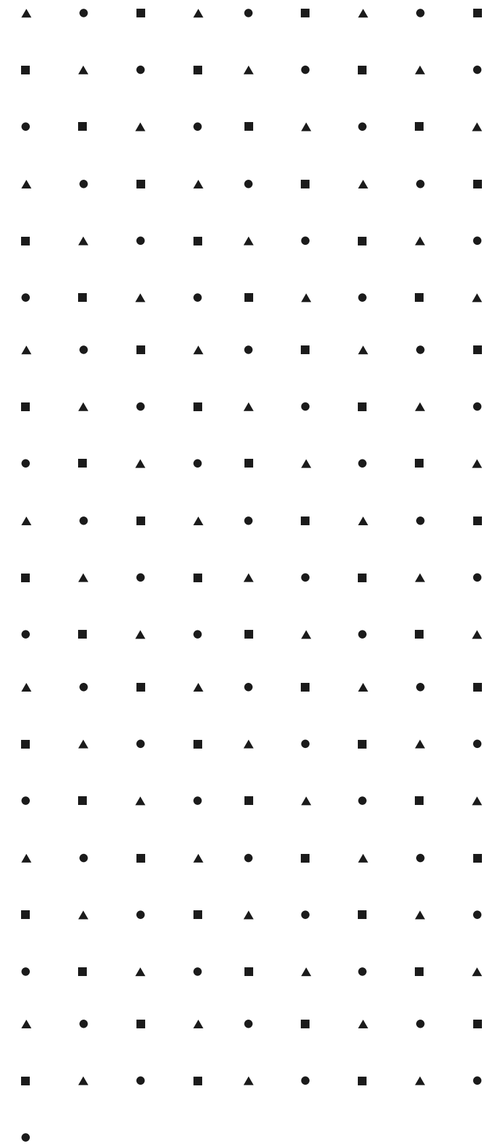
- STAGE/STAGING

  The act of preparing a file to be committed.

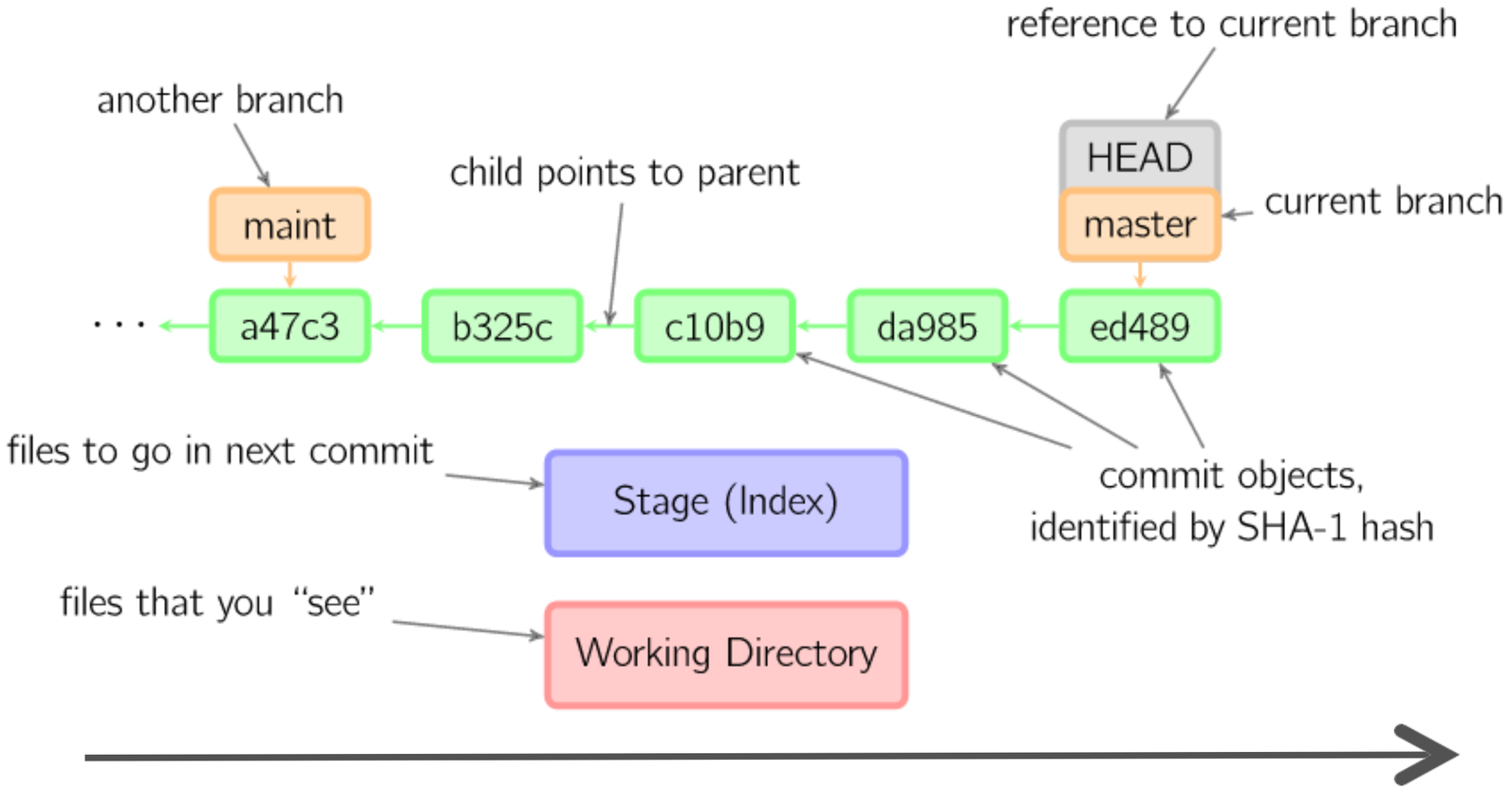
  File are not automatically committed

  AKA Index/Indexing

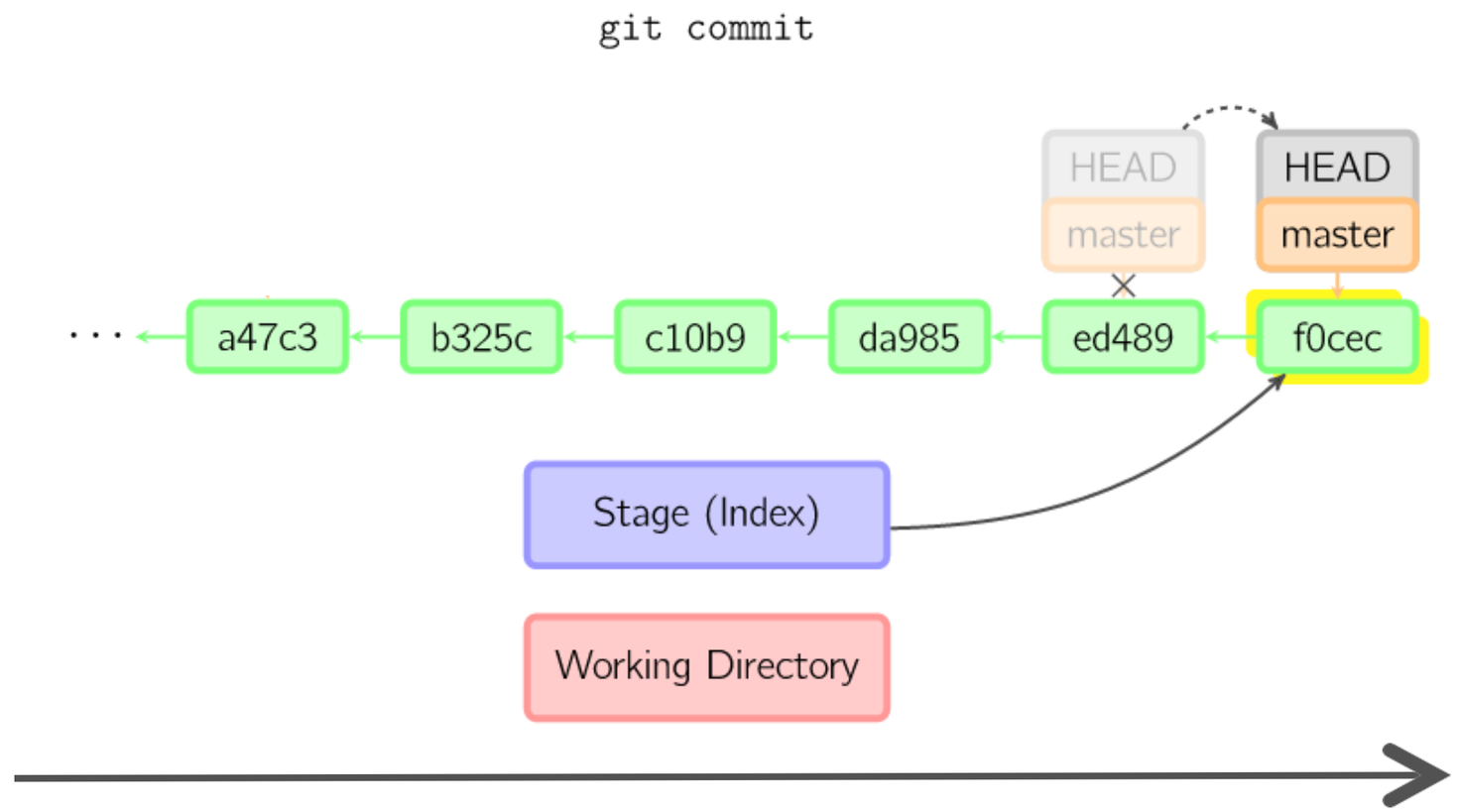
  See: git add/git rm



GIT Stage vs Working



GIT Stage vs Working



The GIT Process

- Use **git pull** to fetch changes from server. **Why???**
- **Optionally** – use git merge and resolve conflicts (hint: see use of local branches)
- Make Changes to files and directories
- Use **git add / rm** to “move” files from Working to Staging (also see **git status**)
- Use **git commit** to prepare a new commit
- Use **git push** to push changes to server (Alternative: Create Pull request and merge)
- Use **git pull** to fetch changes from server

GitHub and Additional Resources

- www.github.com
- Largest web-based git repository hosting service
- Aka, hosts ‘remote repositories’
-
- Allows for code collaboration with anyone online including AI support + extra functionality on top of git
- UI, documentation, bug tracking, feature requests, **pull requests,** *and more!*
- *Enterprise Edition for Business*

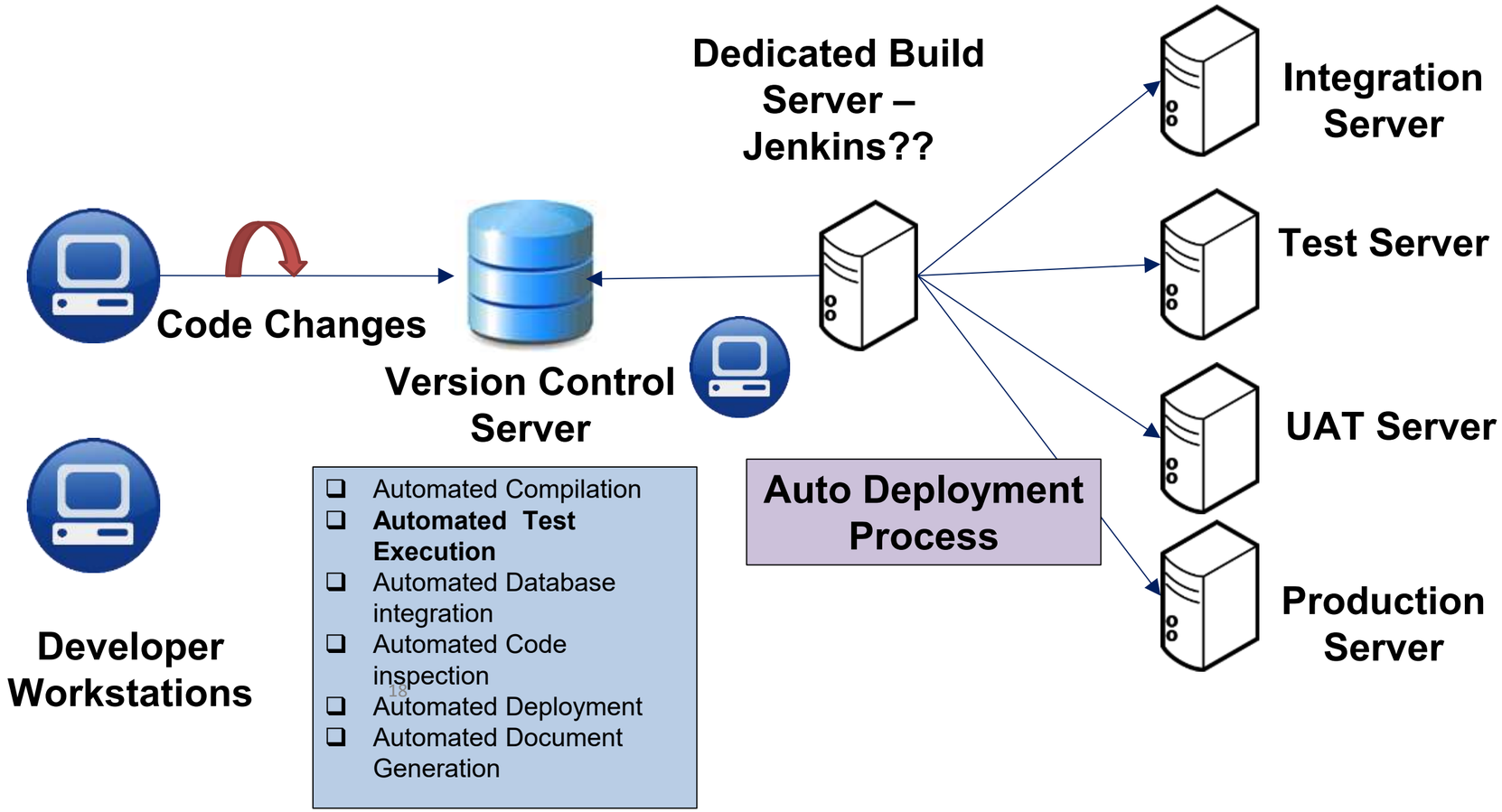
GitHub and Additional Resources

- Official git site and tutorial:
  https://git-scm.com/
- GitHub guides:
  https://guides.github.com/
- Command cheatsheet:
  https://training.github.com/kit/ downloads/github-git-cheat-sheet.pdf
- Interactive git tutorial:
  https://try.github.io/levels/1/challenges/1
- Visual/interactive cheatsheet:
  http://ndpsoftware.com/git-cheatsheet.html

Jenkins and GitHub Actions

A Quick introduction to Jenkins and GitHub Actions

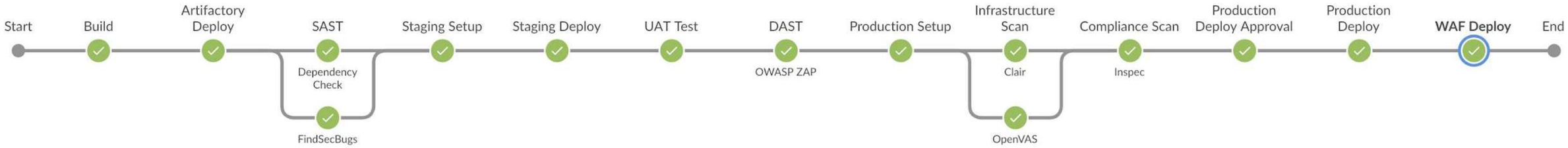
CI/CD pipelines, Jenkins and Actions



CI/CD pipelines, Jenkins and Actions

DevOps Pipeline

DevSecOps Pipeline



CI/CD pipelines, Jenkins and Actions

Bamboo

Hudson

TeamCity

Jenkins

continuum

Visual Studio
Team Foundation Server

CI/CD pipelines, Jenkins and Actions

An open source continuous integration/continuous delivery and deployment (CI/CD) automation software written in the Java programming language.

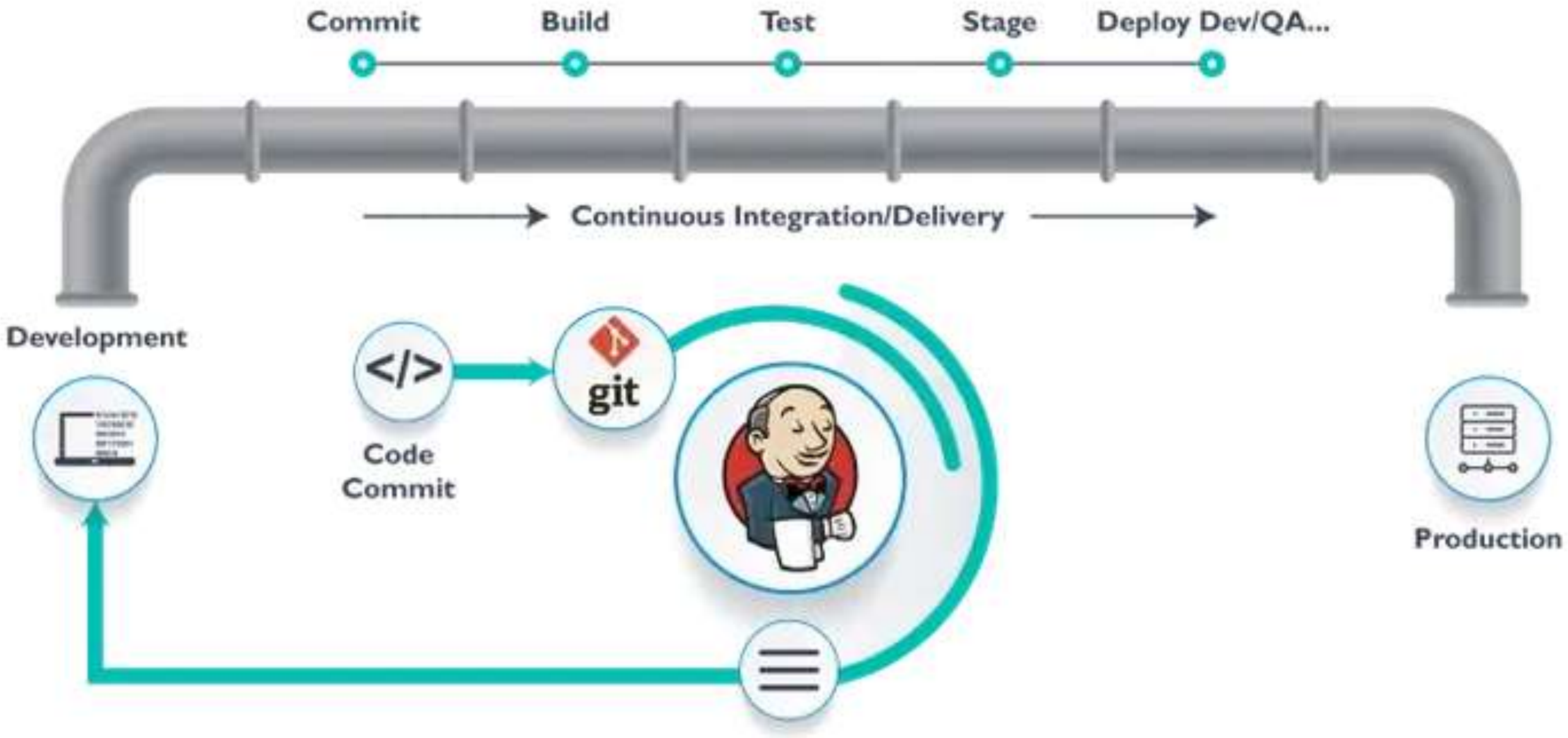
It is used to implement CI/CD workflows, called pipelines.

Jenkins is a self-contained , open source automation that can be used to automate all sorts of tasks related to building , testing and deploying software.

It uses the **plugins** for building and testing the project code continuously .

Jenkins can be installed through native system packages, Docker or even run standalone by an machine with a Java Runtime Environment (JRE) installed.

CI/CD pipelines, Jenkins and Actions



Jenkins Terminology

- JOB — An unit of work for the project
- VIEW — User Defined collection of jobs or a workflow
- MASTER — The Central Jenkins master where jobs are scheduled. Can be more than 1.
- SLAVE — The node where actual work is done. In production, no job is expected to be run on the master

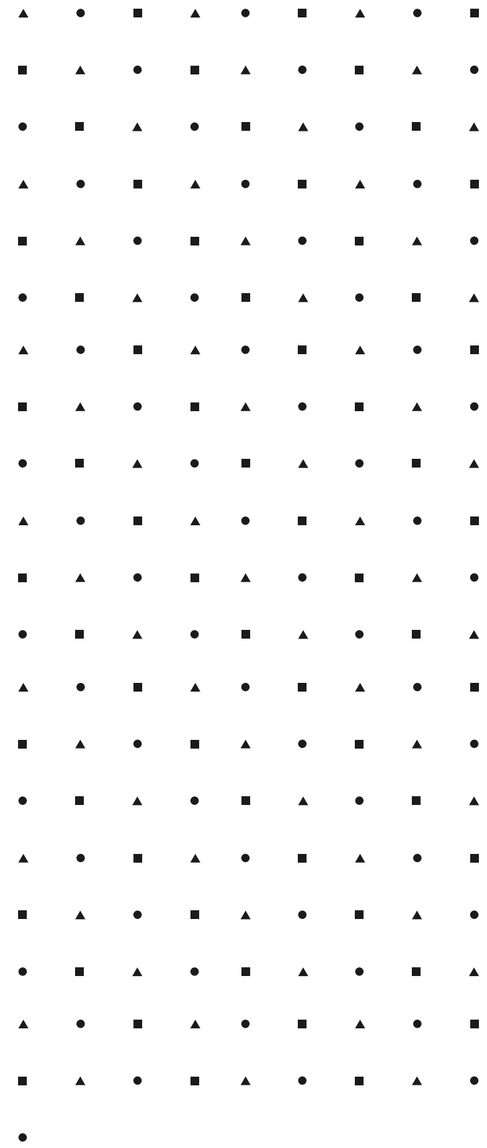
Jenkins Terminology

- WORKSPACE

  The working area where a JOB is carried out.

- PLUGINS

  User or 3rd party SW that actually do the jobs. Over 600+ plugins available.



Jenkins Master Slave Architecture

cismaster

Slaves connect to cismaster over JNLP

jenkins-modslave
Windows XP
Java JDK 6

cisrwin1
Windows 7
R (2.8.1 and upwards)

cisjava1
Centos 5.9
Java JDK 6

cisjava2
Centos 5.9
Java JDK 6

javalinuxslaves

cisjavawin
Windows 7
Java JDK 6

javawindowsslaves

cislave-w7net4
Windows 7
.NET 4

cisr1
Centos 5.9
R (2.8.1 and upwards)

Jenkinsfile

Pipeline as code

Pipelines are expressed as Groovy Domain Specific Language

They are a sequence of steps (including Parallel Execution)

This is a simple file called Jenkinsfile and kept with the actual code (in Version Control System)

Can be

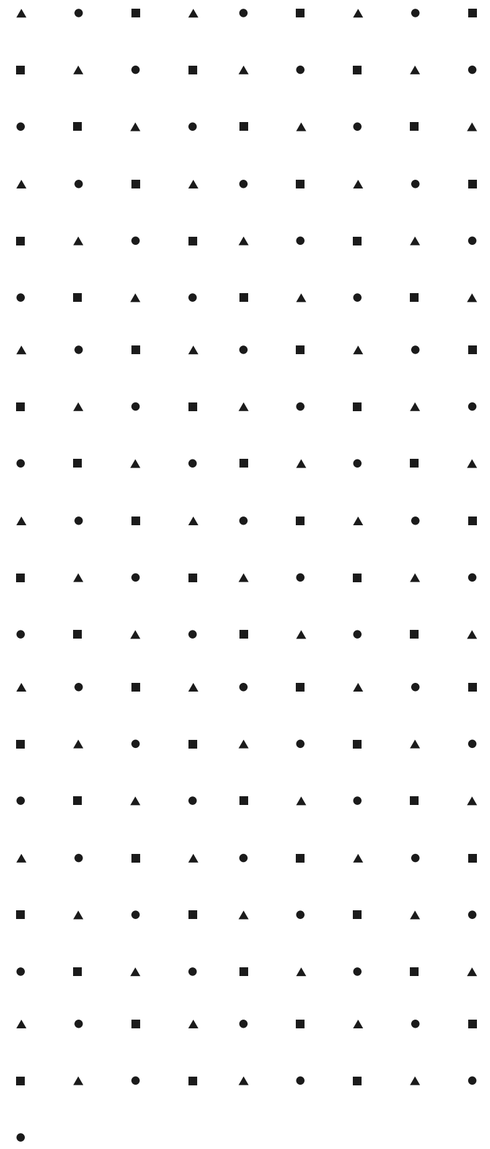
- Declarative
- Scripted

CI/CD pipelines, Jenkins and Actions

```
pipeline {
    agent any

    stages {
        stage('Build') {
            steps {
                echo 'Building..'
            }
        }
        stage('Test') {
            steps {
                echo 'Testing..'
            }
        }
        stage('Deploy') {
            steps {
                echo 'Deploying....'
            }
        }
    }
}
```

```
node {
    checkout scm
    /* .. snip .. */
}
```



GitHub Actions

- Globally available since November 13, 2019
- Based on Azure Pipelines
- Native integration with GitHub API
- YAML-based configuration
- Modular architecture & community-driven
- Runners on Windows, Linux, macOS, or self-hosted
- Available with Free, Pro, Team, Enterprise Cloud, One

GitHub Actions

Free! ✨

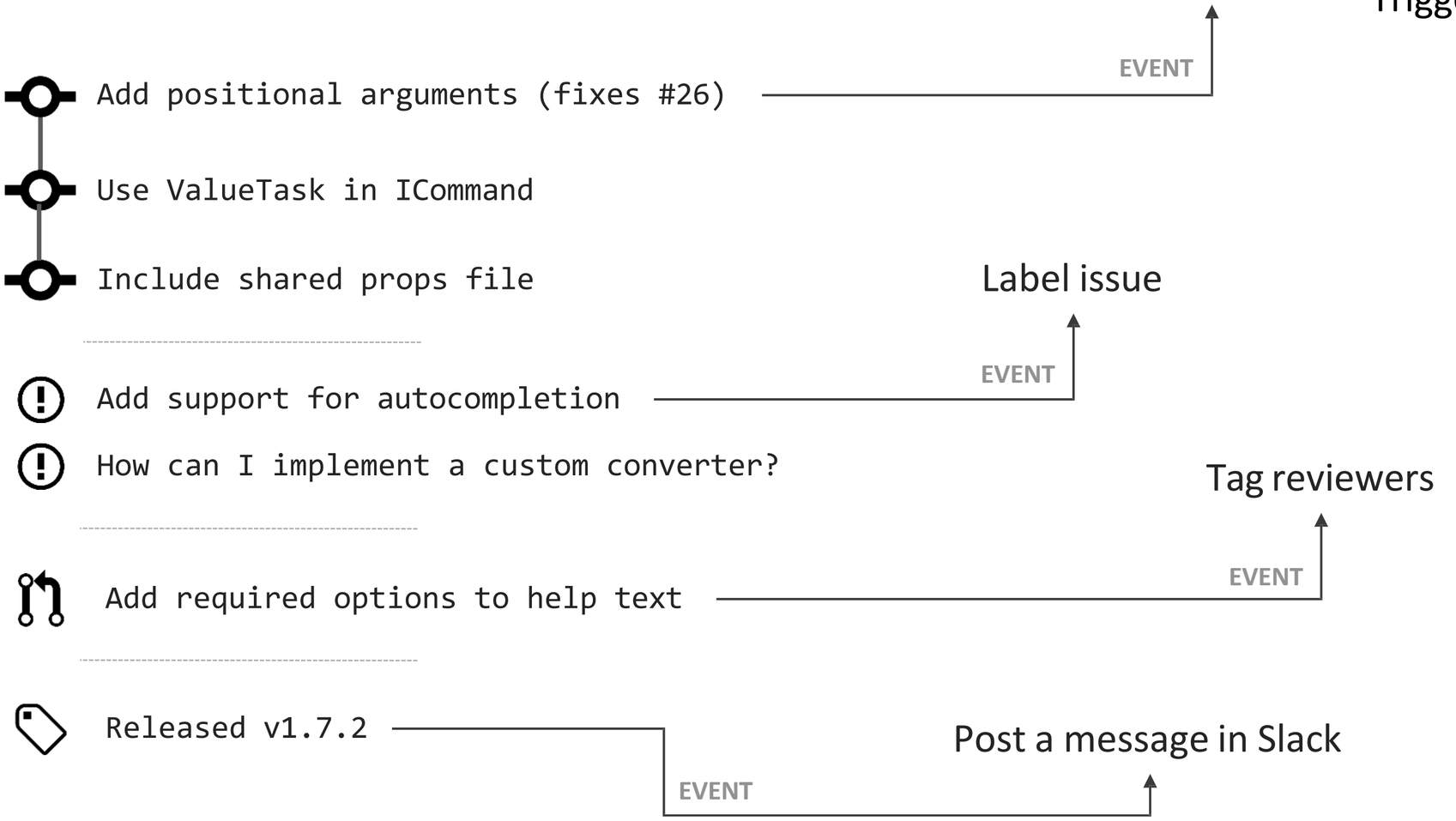
Private repositories

Product	Storage	Minutes (monthly)
GitHub Free	500 MB	2,000
GitHub Pro	1 GB	3,000
GitHub Team	2 GB	10,000
GitHub Enterprise Cloud	50 GB	50,000

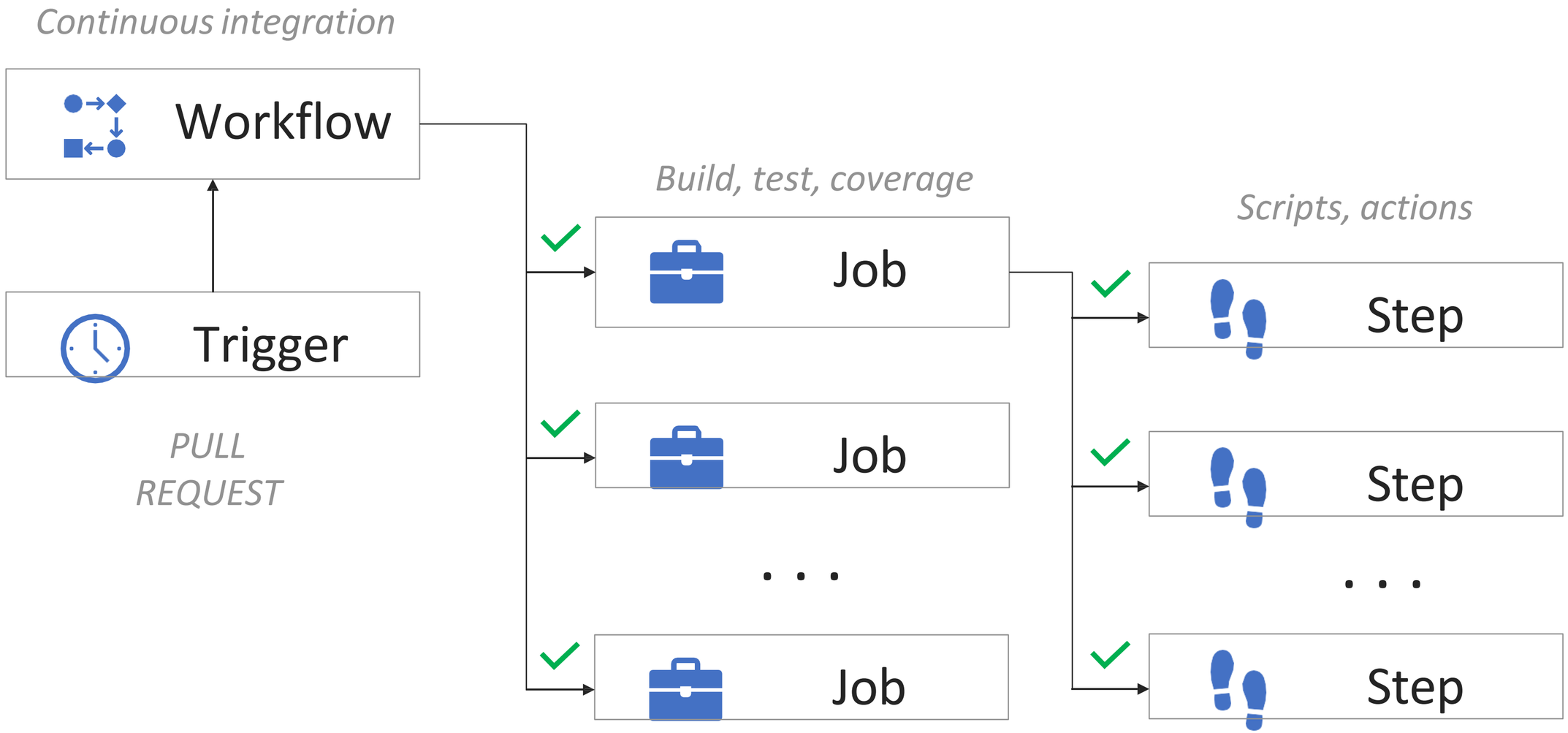
Operating system	Minute multiplier
Windows	2
Linux	1
macOS	10

Operating system	Per-minute rate
Windows	$0.016
Linux	$0.008
macOS	$0.080

GitHub Actions



GitHub Actions



GitHub Actions

~/.github/workflows/CI.ym

```yaml
name: CI
on: [push, pull_request]

jobs:
  build:
    runs-on: windows-latest

    steps:
    - name: Checkout
      uses: actions/checkout@v1

    - name: Install .NET Core
      uses: actions/setup-dotnet@v1
      with:
        dotnet-version: 3.1.100

    - name: Build & test
      run: dotnet test --configuration Release

    - name: Build & publish
      run: dotnet publish LightBulb/ -o LightBulb/bin/Publish/ --configuration Release

    - name: Upload build artifacts
      uses: actions/upload-artifact@master
      with:
        name: LightBulb
        path: LightBulb/bin/Publish/
```

Trigger on new commits and pull requests

Clone repository and checkout HEAD
(commit hash is passed as env var)

Install .NET Core v3.1.100

Run custom shell scripts

Upload specified directory as
a ZIP artifact

GitHub Actions Triggers

- **GitHub API events** ⚡
  `push`, `pull_request`, `issues`, `release`, and 20 others
- **Schedule** 🕒
  Cron syntax, e.g.: `*/15 * * * *`
- **Manual** 🌐
  POST to /repos/:owner/:repo/dispatches

GitHub Actions

```yaml
# Trigger on push events on s
pecific branches
on:
  push:
    branches:
      - 'master'
      - 'release/*'
```

```yaml
# Trigger on manual dispatch
on: repository_dispatch
```

```yaml
# Trigger every midnight UTC
on:
  schedule:
    - cron: '0 0 * * *'
```

```yaml
# Trigger when an issue is opened o
r labeled
on:
  issues:
    types: [opened, labeled]
```

GitHub Actions Referencing actions

- By GitHub repository
  `{owner}/{repo}@{ref}`
  `{owner}/{repo}/{path}@{ref}`

  `jessfraz/branch-cleanup-action@master`
  `johndoe/my-actions/push-image@v1`
- By file path
  `./path/to/dir`

  `./.github/actions/my-action`
- By Docker image
  `docker://{image}:{tag}`

  `docker://hello-world:latest`

GitHub Actions Docker containers

```yaml
jobs:
  build:
    runs-on: ubuntu-latest

    services:
      redis:
        image: redis
        ports:
        - 6379/tcp
        options: --entrypoint redis-server

    steps:
      - uses: actions/checkout@v1

      - run: node client.js
        env:
          REDIS_HOST: localhost
          REDIS_PORT: ${{ job.services.redis.ports[6379] }}
```

Image from the Docker registry

Bind port 6379 in container to a random port on host

Custom arguments passed to `docker create`

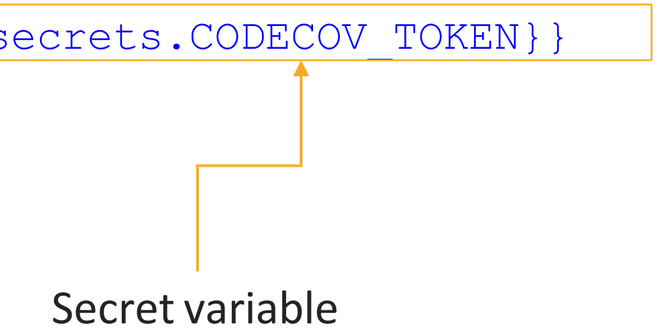
Exposed port is resolved dynamically

GitHub Actions Secrets

```yaml
steps:

# ...

- name: Collect coverage report
  run: |
    choco install codecov --no-progress
    codecov -f LtGt.Tests/bin/Release/Coverage.xml -t ${{secrets.CODECOV_TOKEN}}
```



GitHub Actions Conditionals

```yaml
jobs:
  build:
    runs-on: ubuntu-18.04

    steps:
    - uses: actions/checkout@v1
    - uses: actions/setup-dotnet@v1
      with:
        dotnet-version: 3.1.100

    - run: dotnet test src
    - run: dotnet pack src

    - if: github.event_name == 'push' && startsWith(github.ref, 'refs/tags/v')
      run: dotnet nuget push src/**.nupkg -k ${{secrets.NUGET_TOKEN}}
```

Conditional expression

@Tyrrrz

GitHub Actions Things we can do with GitHub Actions

- Run tests on every commit
- Publish Docker image when a tag is pushed
- Label an issue by content when it’s created
- Run nightly E2E tests
- Automatically close stale issues every week
- Invite new contributors to sign the CLA when a PR is opened
- Automatically format code on push
- …

@Tyrrrz

GitHub Actions Summary

- GitHub Actions is an automation platform (not just CI/CD)
- Can trigger workflows on various events
- Workflows are based on actions which are sourced by the community
- Free for all public repos, pay-as-you-go for private repos
- Easy to set up and configure to your needs

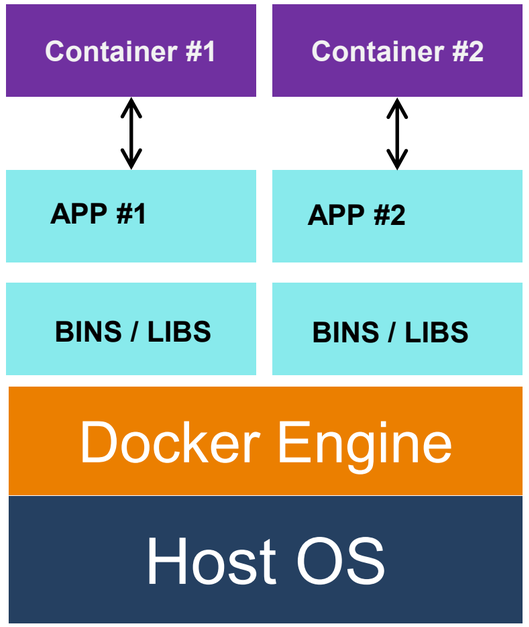
GitHub Actions For the curious

- Awesome Actions by Sarah Drasner
  https://github.com/sdras/awesome-actions
- GitHub Actions Advent Calendar by Edward Thomson
  https://edwardthomson.com/blog/github_actions_advent_calendar.html
- Comprehensive Introduction to GitHub Actions by Tierney Cyren
  https://dev.to/bnb/an-unintentionally-comprehensive-introduction-to-github-actions-ci-blm
- Official documentation
  https://help.github.com/en/actions

Docker and Containers

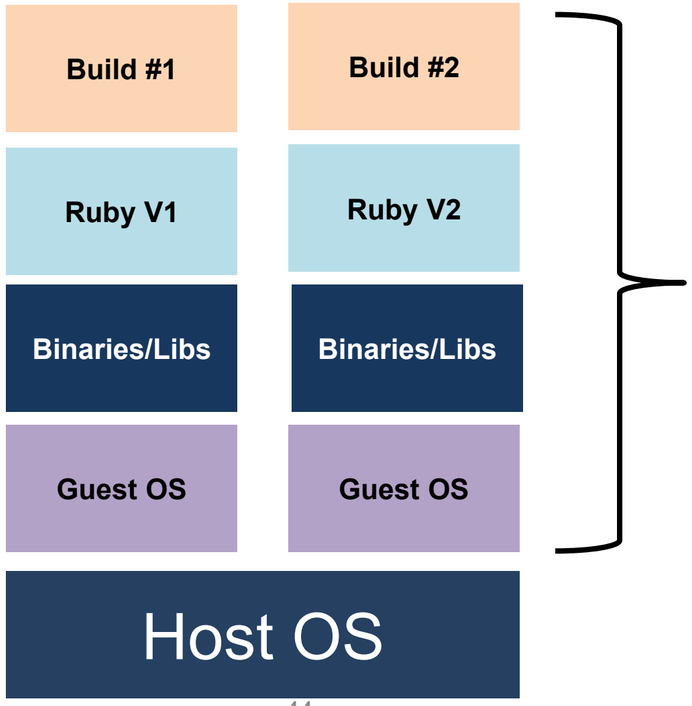
A Quick introduction to Docker and Containers

Docker and Containers

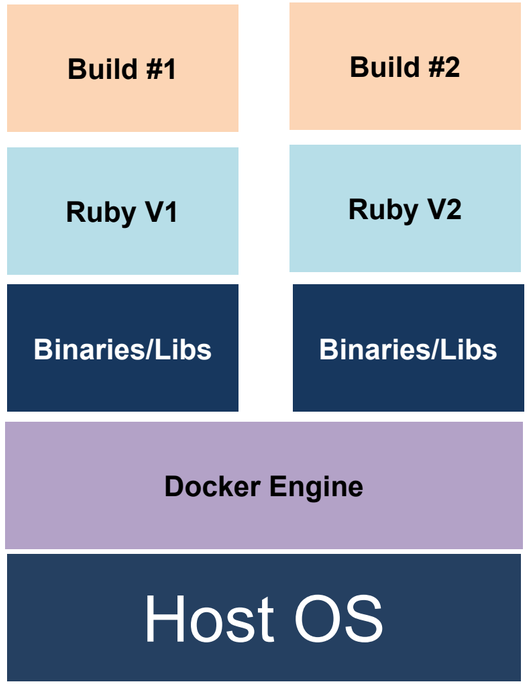


- A Containerization Platform
- All Apps and Library dependencies packed as one unit.
- One Image – One Task Principle.
- Best Suited for Microservices
- **Core of Docker is the Docker Engine**

Docker and Containers



Virtual Machines

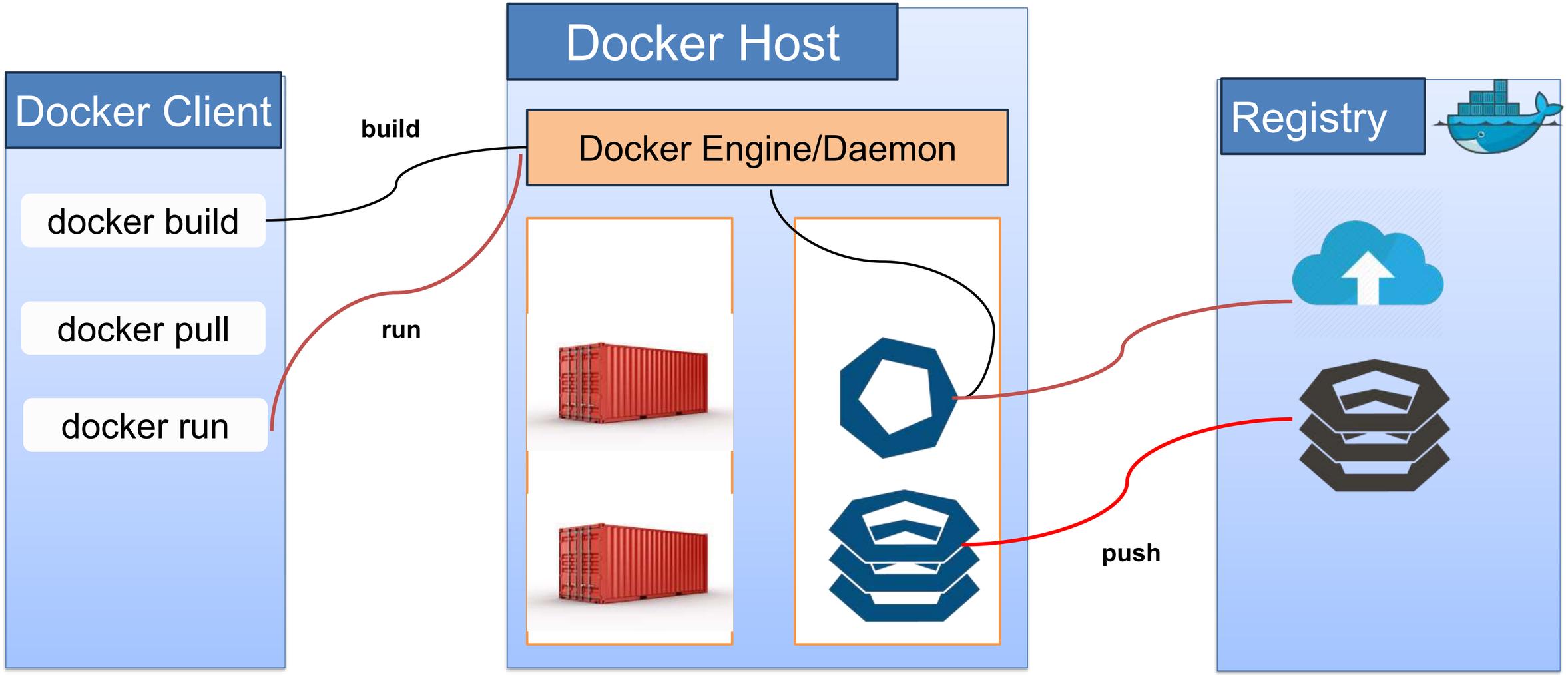


Containers

Docker and Containers

- **Consistent and Isolated Environment**
- Better Resource Utilization
- Rapid Application Deployment
- Deploy Immutable Infra Faster than physical or virtual machines
- Faster start up times & Faster Continuous Delivery – Make Multiple environments available really fast.
- Reliable Continuous Delivery – Same Binary Deployed in same environment
- Can Roll back image versions ☺

Docker Architecture



Docker Terminology

- IMAGES

  A template that contains instructions on running applications
  See: **docker images**/**docker rmi**

- CONTAINERS

  A running instance of an image
  See: **docker run** /**docker ps**/**docker rm**

- DOCKERFILE

  A Text file that contains instructions to create a new image.
  See: **docker build**

- VOLUMES

  Volumes are locations to persist data after containers are stopped/removed

Docker Terminology

- PORTS

  A Communication endpoint to send and receive data

- NETWORK

  A network is a collection of nodes/containers. Docker has its own networking rules, though we largely work with **bridged** networks.

- CONTEXT

  Each Daemon (Docker Engine) represents an isolated set of resources called a context

Basic Docker Commands

- docker build

docker build –t myimage .
Creates an image called myimage using a file called Dockerfile and accessing content in current directory (.). The . here is called as a build context.

- docker run

docker run myimage –p 85:80
Runs an image myimage exposing port 80 as port 85 on local machine.

- docker ps

docker ps
Lists containers running.

Basic Docker Commands

- docker rm

docker rm <id|name>
Removes a container referred either by id or name. A running container cannot be removed. Stop it first using ***docker stop***

- docker rmi

docker rmi <id|name>
Removes an image. If an image has containers, it cannot be removed. Remove containers before removing image

- docker logs

docker logs <id|name>
Show the logs for the container

Docker Commands

Docker commands have options.

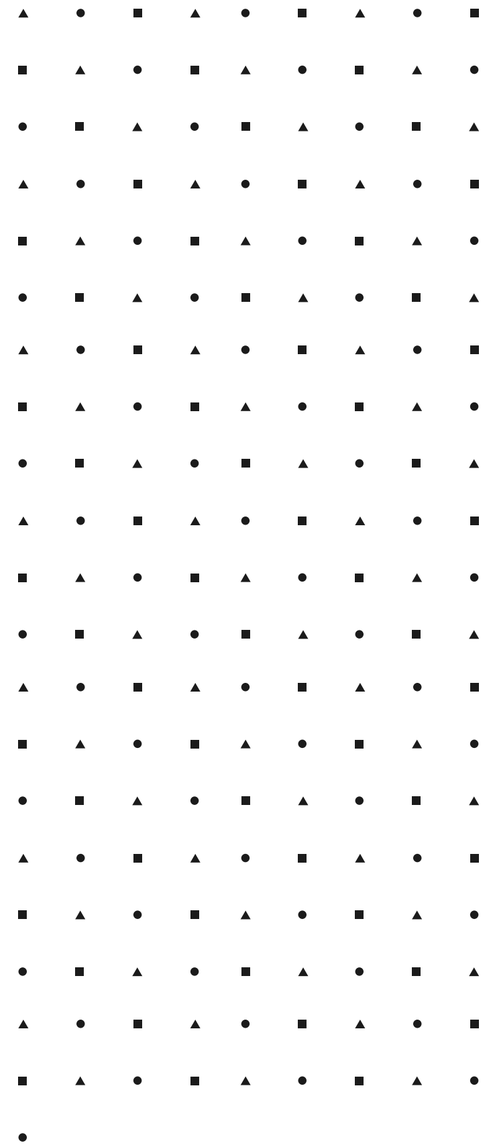
e.g. docker run –itd …….

In the above,

-itd is a short form for –i –t –d

And is used to

Allow for interactions via keyboard (-i)
Allow for interactions via terminal (-t)
Run in background (-d)



Docker Commands

Resources

[The Docker CLI Cheat Sheet]

[The Ultimate Docker Cheat Sheet]

Docker Builds using DockerFiles

A Dockerfile contains instructions to create an image.

It is a plain text file

```
FROM ubuntu:latest
RUN echo “Hello World”
```

All docker images are either built from

- Pre-existing images
- Scratch

Docker Builds using DockerFiles


```
FROM ubuntu:latest
CMD [“echo”, “Hello World”]
```

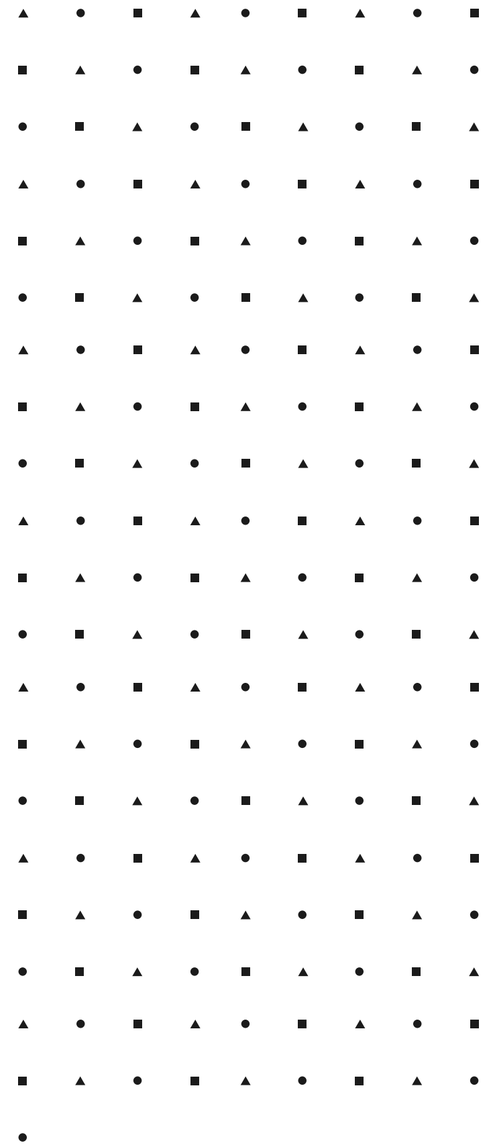
Notice the difference between the previous example.

In the first case, nothing happens when you run it.
In the above, a container is started, Hello World is printed and container stopped.

In both cases, you have a temporary ubuntu machine running

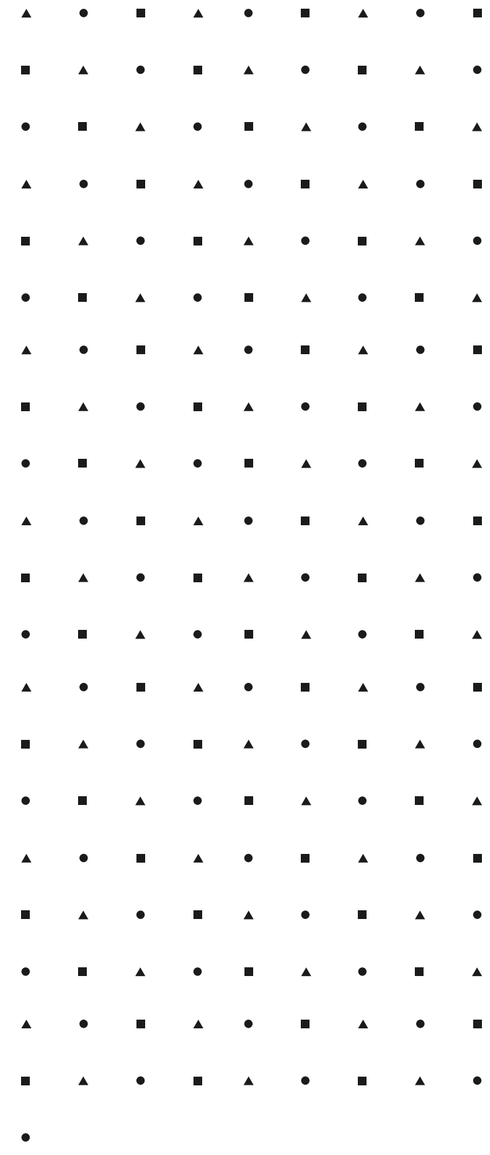
Docker Builds using DockerFiles

• FROM	Specify the base image to be used. It could be scratch and could have more than one FROM directive.
• ENV	Sets Env variables used for the Building the images
• ARG	Runtime arguments for the build
• CMD / ENTRYPOINT	The starting point for the image. CMD is used to parametrize the starting program.



Docker Builds using DockerFiles

- EXPOSE — Make Ports available to the outside world
- USER — User ID to run inside the container
- WORKDIR — The working directory to use inside the container
- VOLUME — Share data between host and container
- RUN — Execute multiple commands during build. Used to customize the image



Docker Builds using DockerFiles


```
FROM ubuntu:latest
ENTRYPOINT [“/bin/echo”, “Hello”]
CMD [“World”]
```

Notice the difference between the previous example.

Now run it as:

docker run –it hello (and)
docker run –it hello Sriram

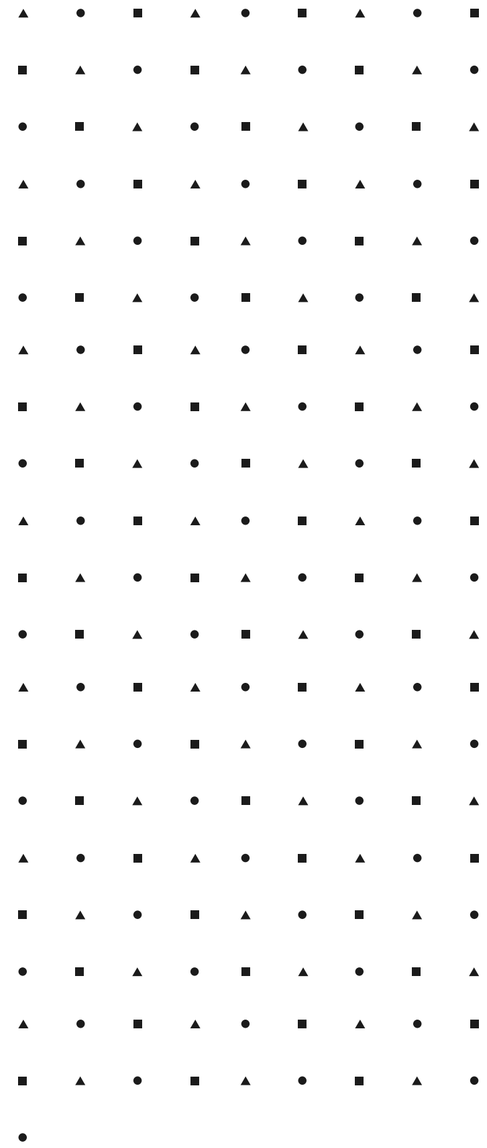
Docker Builds using Docker compose

Docker files are used to **create** single images

Images are run using docker run.

In complex applications, there is a need to orchestrate multiple containers.

A better option is to use **Docker compose** to specify how these applications (Services) relate to each other and co-ordinate both building and running of containers.



Docker Builds using DockerFiles

Resources

Docker File Reference

Dockerfiles on Windows

The Ultimate Docker Cheat Sheet

Installation and Demo

Installation

GIT

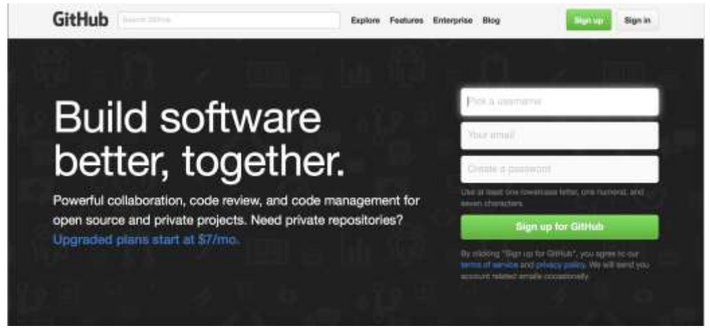
Docker

Jenkins

Install GIT

- **Linux (Debian)**
  -Command: `sudo apt-get install -y git`
- **Linux (Fedora)**
  -Command: `sudo yum install -y git`
- **Mac**
  -http://git-scm.com/download/mac
- **Windows**
  -http://git-scm.com/download/win

- www.github.com
- Free for public repositories



Configure your account to use SSH Keys.

Use git config to set user.name and user.email

```
git config -global user.name <yourname>
git config -global user.email <youremail>
```

Install Docker

Docker for windows can be installed via Docker Desktop

https://desktop.docker.com/win/main/arm64/Docker%20Desktop%20Installer.exe?utm_source=docker&utm_medium=webreferral&utm_campaign=dd-smartbutton&utm_location=module

Install Jenkins

The Jenkins installer for Windows is a single step.

You will need to install Java first.

The installer can be downloaded here.
https://www.jenkins.io/download/

Choose the LTS version (https://www.jenkins.io/download/thank-you-downloading-windows-installer-stable)

Install Jenkins

Alternatively, if you have git and Docker installed, clone

https://github.com/seshagirisriram/Devops

Navigate to docker\Jenkins and run

install_jenkins.bat (windows)
install_jenkins.sh (Unix)

Navigate to http://localhost:8080 and finish the Jenkins setup.

NB: You will need to add SSH keys to the Jenkins server and GitHub server to automate the CI/CD process.