



WORKFORCE DEVELOPMENT

**Configuration Management and DevOps:
Automate Infrastructure Deployments**

PARTICIPANT GUIDE



Content Usage Parameters

Content refers to material including instructor guides, student guides, lab guides, lab or hands-on activities, computer programs, etc. designed for use in a training program

1

Content is subject to
copyright protection

2

Content may only be
leveraged by students
enrolled in the training
program

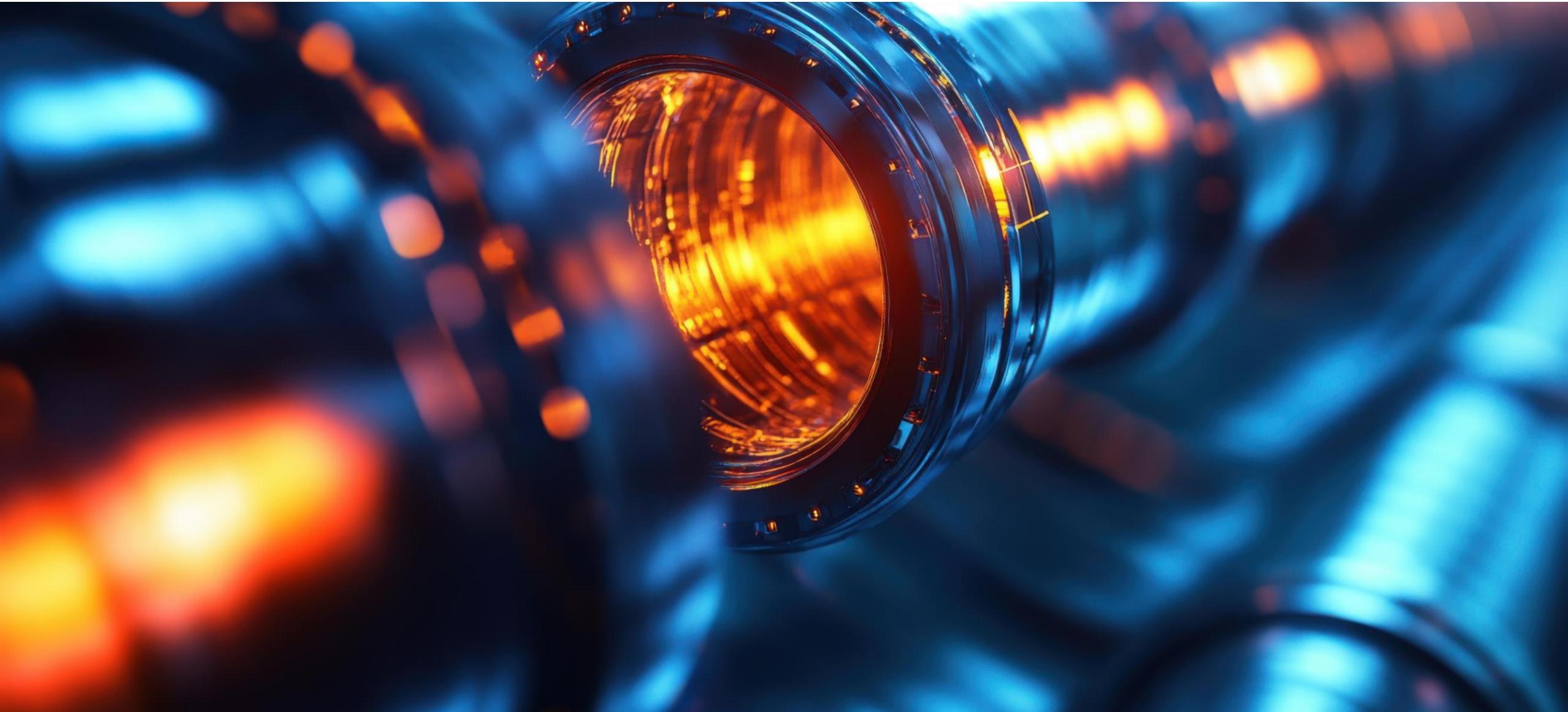
3

Students agree not to
reproduce, make
derivative works of,
distribute, publicly perform
and publicly display
content in any form or
medium outside of the
training program

4

Content is intended as
reference material only to
supplement the instructor-
led training

DevOps Foundation



Logistics



- Class Hours:
- Instructor will provide class start and end times.
- Breaks throughout class



- Telecommunication:
- Turn off or set electronic devices to vibrate
- Reading or attending to devices can be distracting to other students

- Miscellaneous
 - Courseware
 - Bathroom

Course Objectives

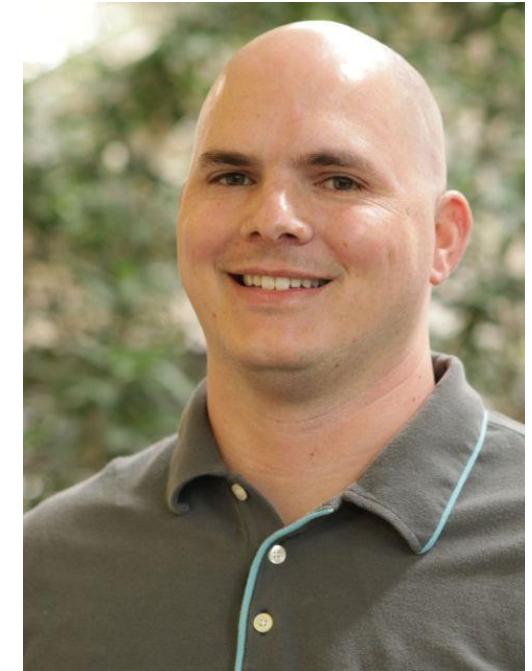


- Explain the benefits of DevSecOps
- Use Git for version control
- Describe what Configuration Drift is and how to avoid it using Infrastructure as Code
- Build pipelines using Azure DevOps to automate infrastructure and application deployment.

Hi!

Jason Smith

Cloud Consultant with a Linux sysadmin background.
Focused on cloud-native technologies: automation,
containers & orchestration



LinkedIn

<https://www.linkedin.com/in/jruels/>

mail

jason@innovationinsoftware.com

github

<https://github.com/jruels>

Expertise

- Cloud
- Automation
- CICD
- Docker
- Kubernetes

Introductions

Hello!

- Name
- Job Role
- Your experience with (scale 1 – 5):
 - Git
 - Infrastructure-as-Code
 - Cloud
 - Azure DevOps
- Expectations for course (please be specific)

<https://jruels.github.io/az-devops>



Traditional Org Structure



POP QUIZ: Organization



How is your company
organized?

Arranged by Functional Area

Research & Development

Manufacturing

Logistics

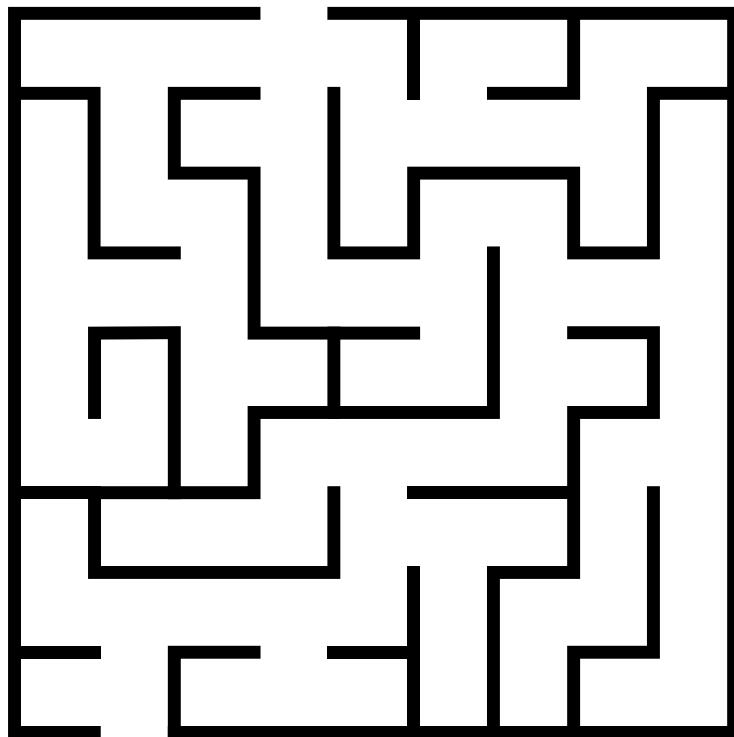
I.T.

Sales &
Marketing

Human Resources

Legal & Finance

Complex Deployment Pipeline



- Many handoffs
- Numerous teams involved
- Manual intervention
- Downtime expected (planned for)

Software Development Life Cycle



POP QUIZ: Software Development Life Cycle



What is your organization's process for developing and deploying software/releases?

Dev Ops & QA

OLD WAY:

CODE →



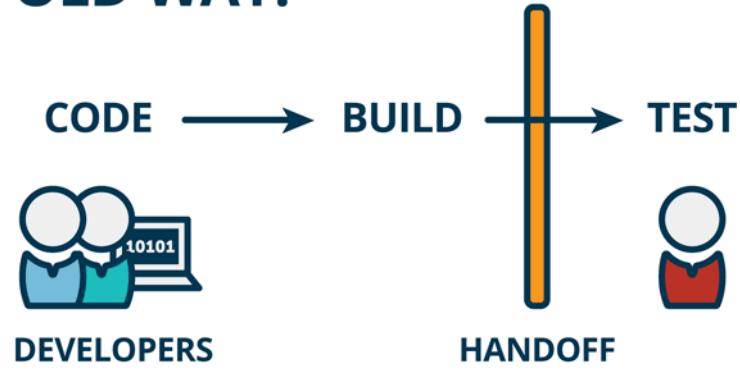
DEVELOPERS

Developers build application

- Runtime
- Libraries
- Dependencies

Dev Ops & QA

OLD WAY:

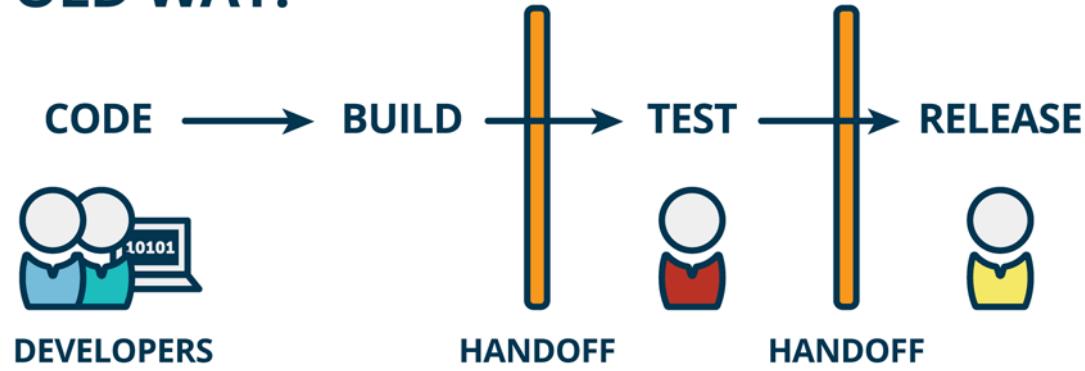


Handed off to Test

- Unit tests
- Integration tests
- Was story completed?

Dev Ops & QA

OLD WAY:

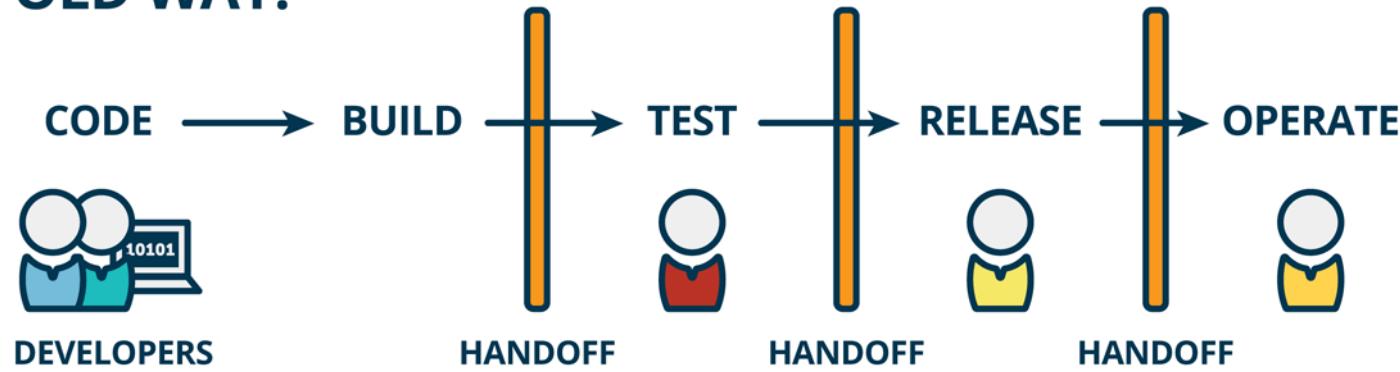


Deployed to Production

- Deployment tooling
- Artifacts
- Services

Dev Ops & QA

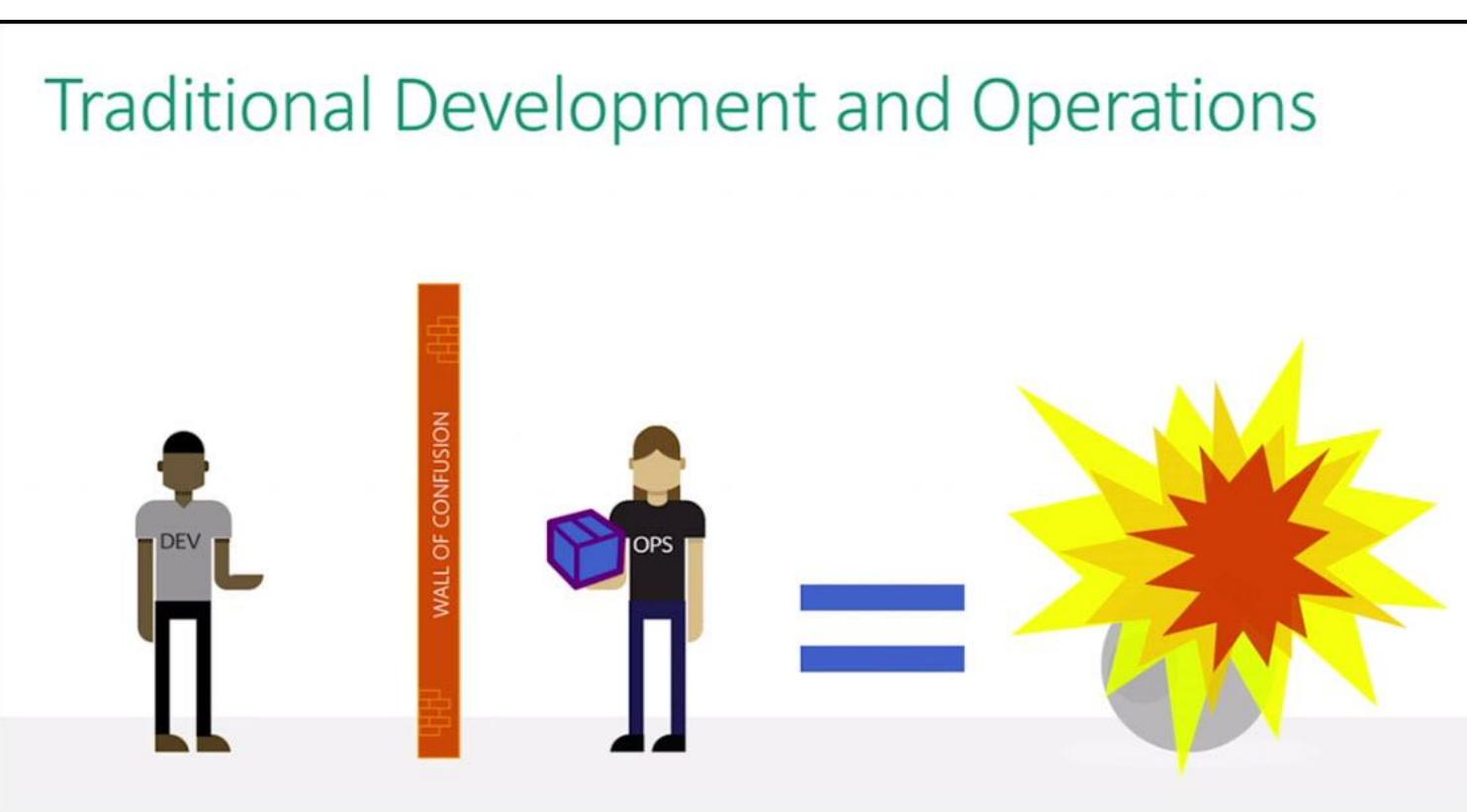
OLD WAY:



Ops now manages applications

- New releases introduce change
- Uptime
- Storage
- etc.

Dev vs Ops



What really happens

OLD WAY:

CODE →



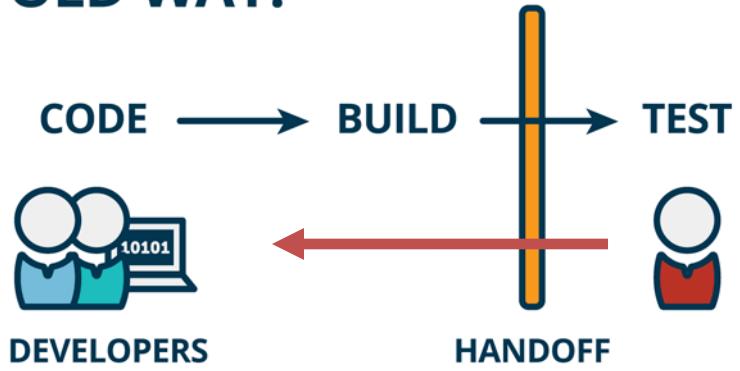
DEVELOPERS

Developers build application

- Runtime
- Libraries
- Dependencies

What really happens

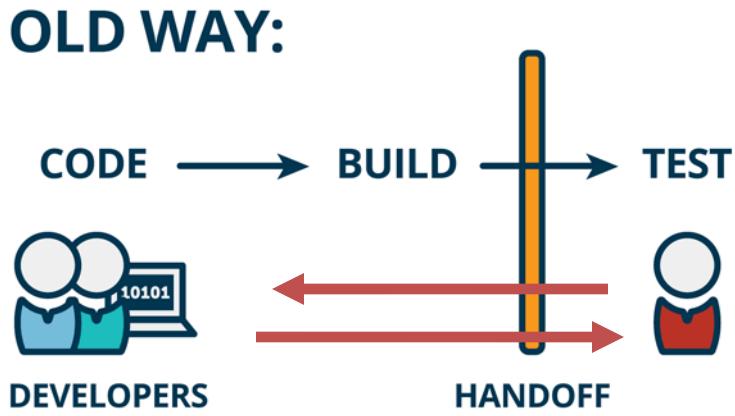
OLD WAY:



Handed off to Test

- Unit tests
- Integration tests
- Was story completed?

What really happens

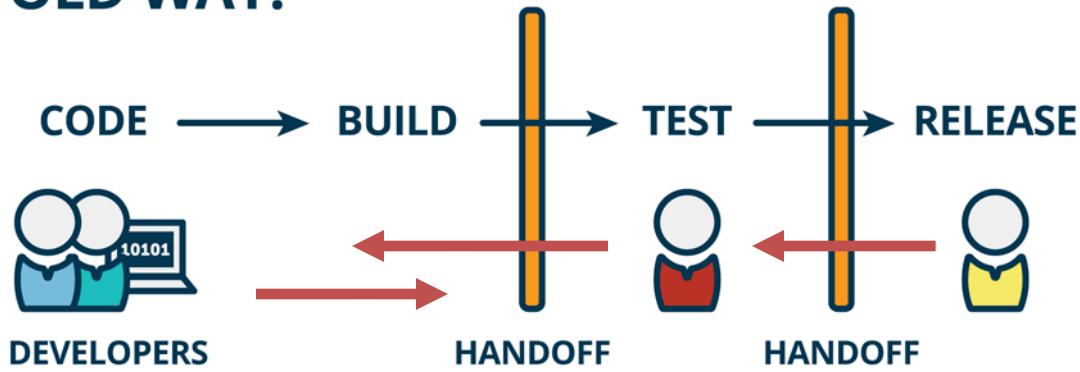


Handed off to Test

- Unit tests
- Integration tests
- Was story completed?

What really happens

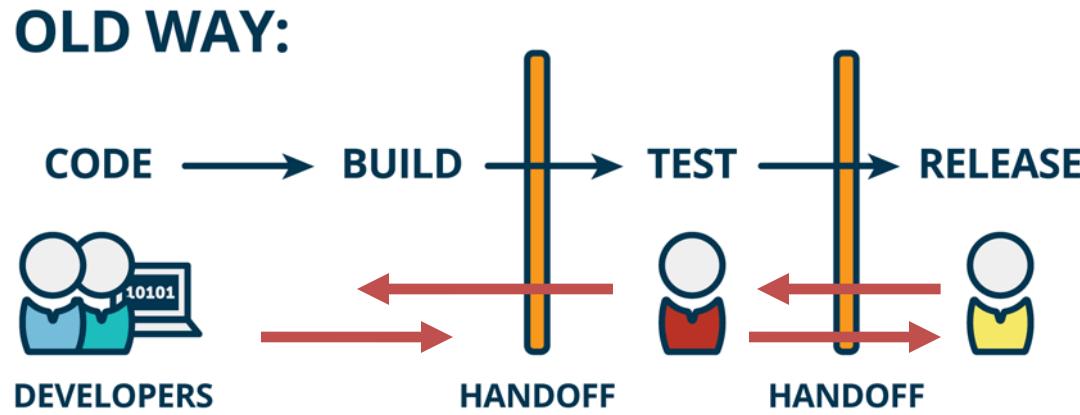
OLD WAY:



Deployed to Production

- Deployment tooling
- Artifacts
- Services

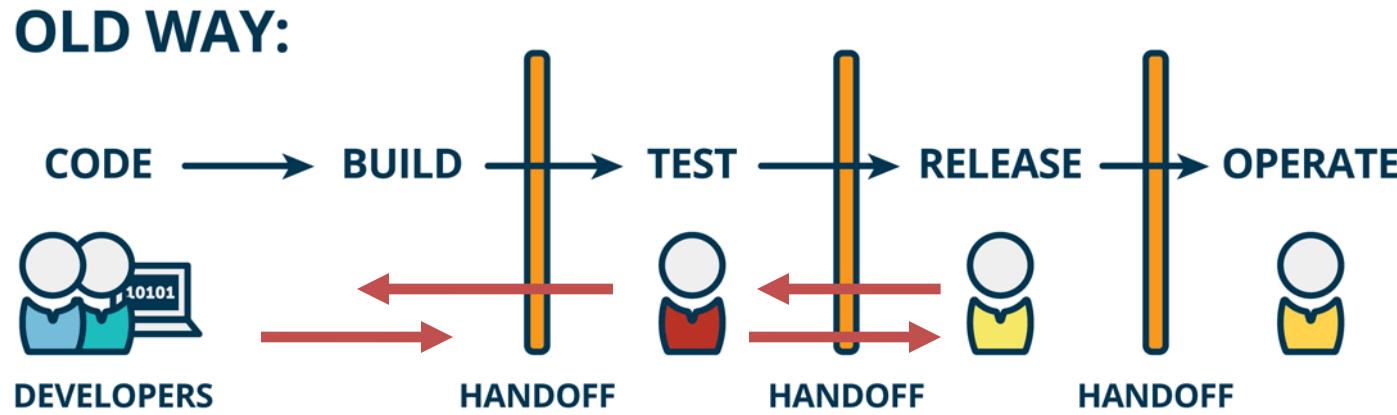
What really happens



Deployed to Production

- Deployment tooling
- Artifacts
- Services

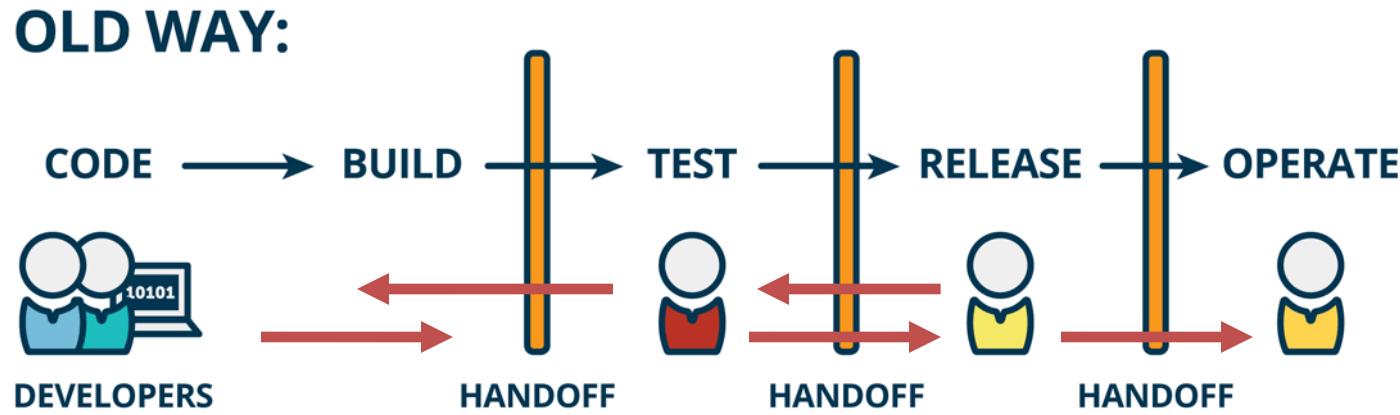
What really happens



Ops now manages applications

- New releases introduce change
- Uptime
- Storage
- etc.

What really happens



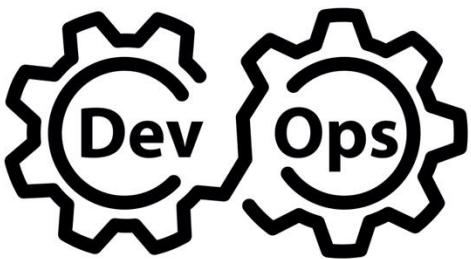
Ops now manages applications

- New releases introduce change
- Uptime
- Storage
- etc.

What is DevOps?



C.A.L.M.S



- Culture
- Automation
- Lean
- Metrics
- Sharing

Culture:

People and process at the forefront.



POP QUIZ: Developers responsibilities?



As a developer what are your primary responsibilities?

People - Developers



- Dev
 - Create
 - Innovate
 - As much change as possible!

POP QUIZ: Operations responsibilities?



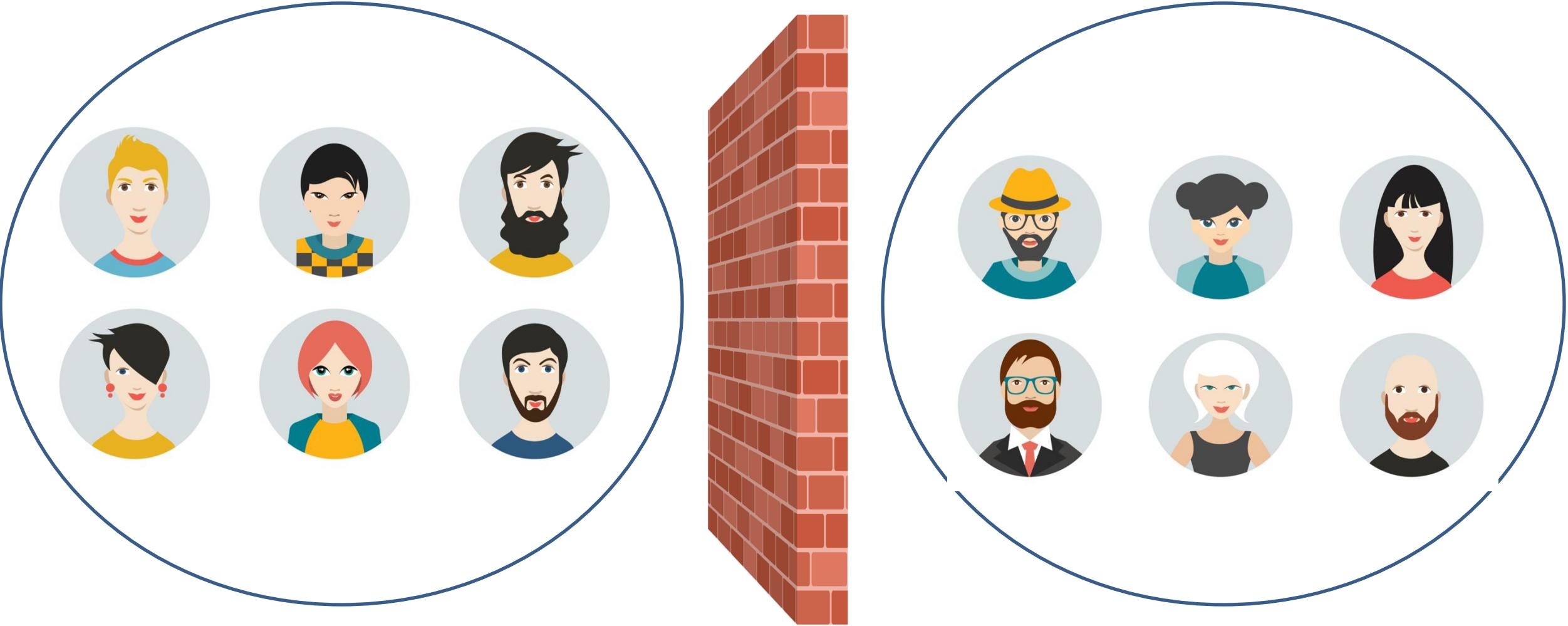
As an operations engineer
what are your primary
responsibilities?

People - Operations

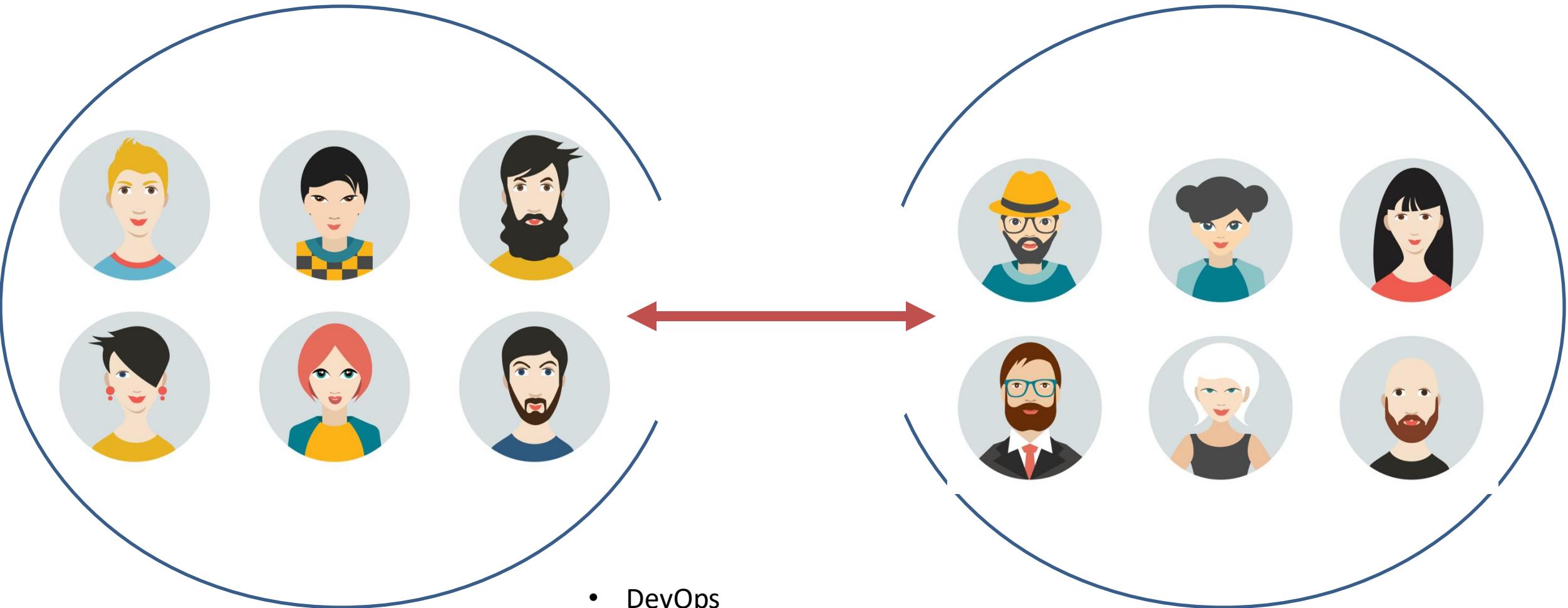


- Ops
 - Stability
 - Deliver services
 - Resist change!

Silos



Collaboration!

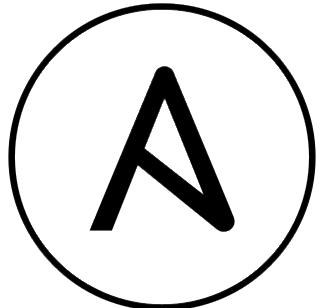


Automation:

Repeatability by automating activities.



Automation tools



ANSIBLE



CHEF



puppet

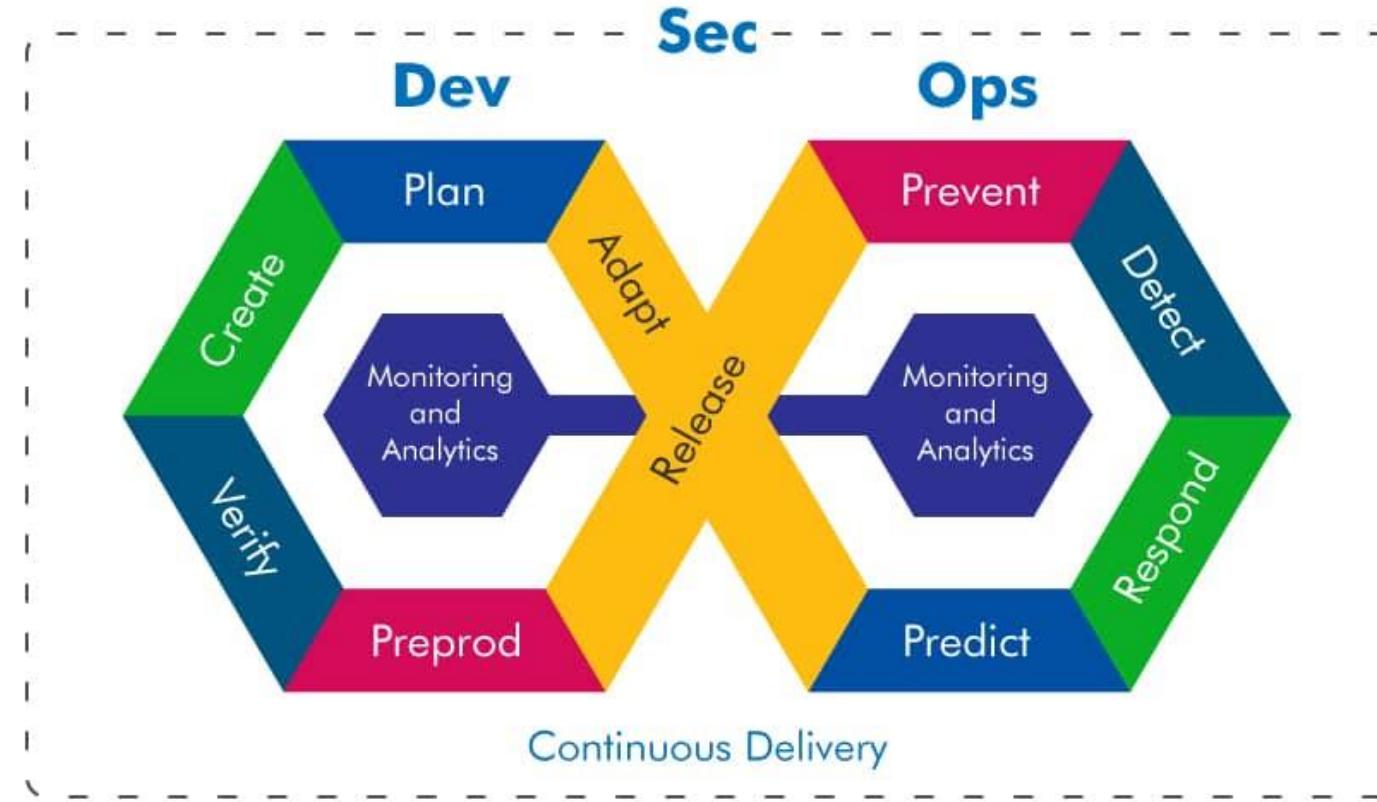


HashiCorp
Terraform

DevSecOps and Policy-as-Code



Introduction to DevSecOps



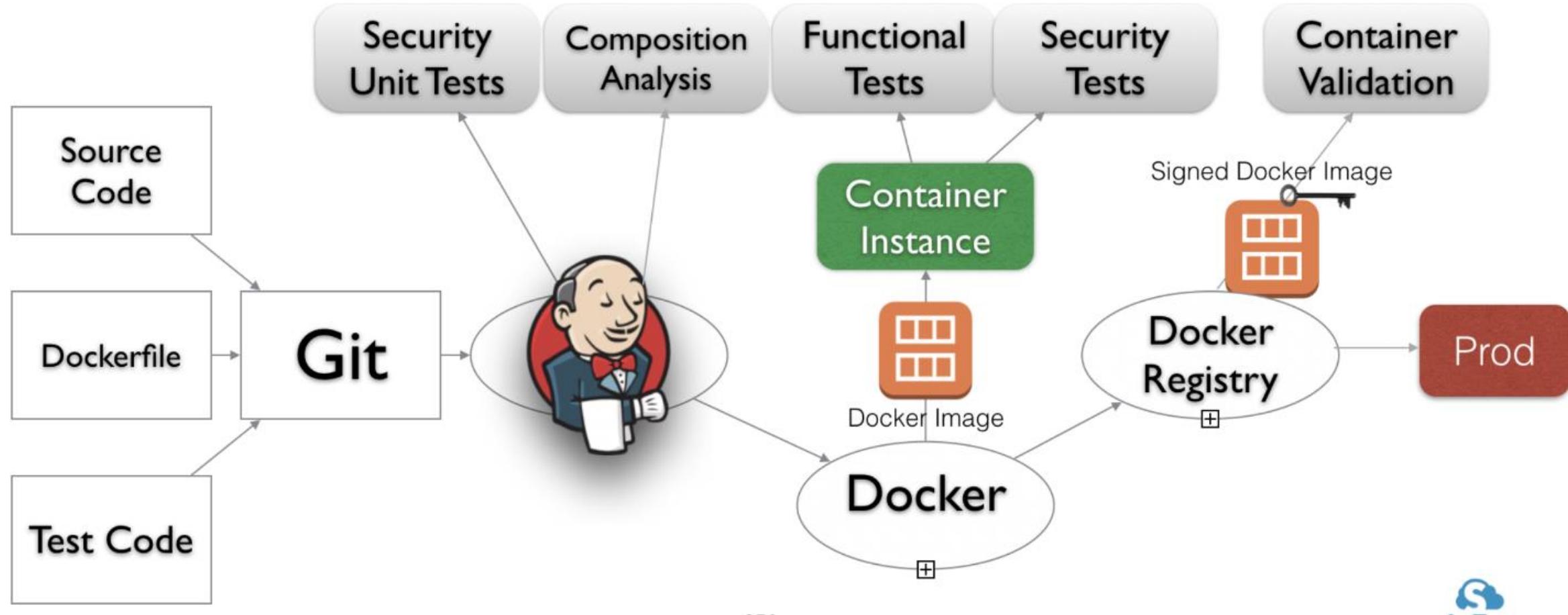
Continuously assessing security measures within the development cycle to gauge their effectiveness and foster ongoing improvement.

What is DevSecOps?

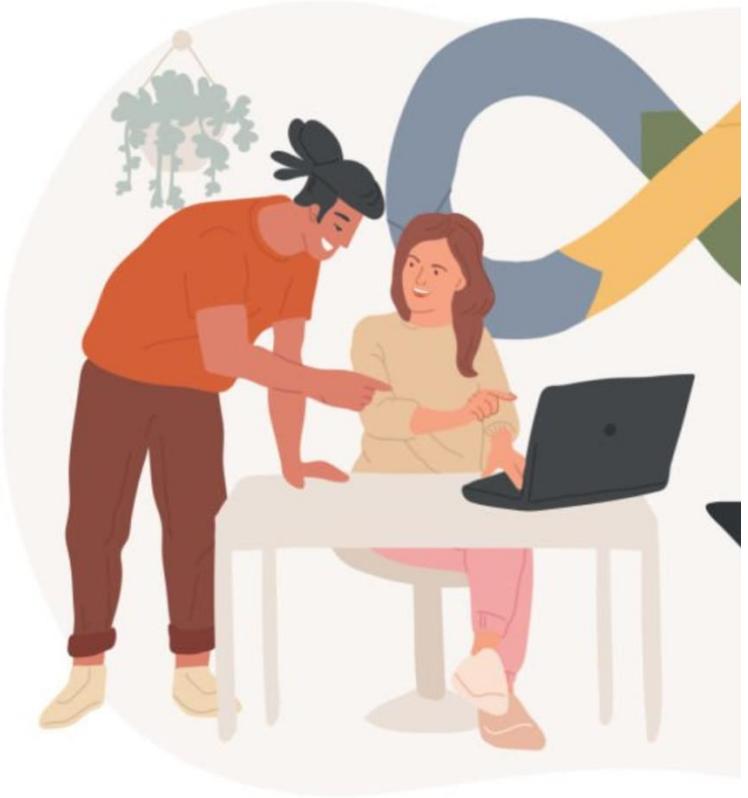


- **DevSecOps** integrates security practices into DevOps, making security a shared responsibility throughout the software development lifecycle.
- **Security** is embedded at every stage, from code development to deployment.
- **Goal:** Catch vulnerabilities early and ensure continuous protection.

Example DevSecOps Architecture



Key principles of DevSecOps



- **Continuous security integration**, automating security checks for faster, safer releases.
- **Close collaboration** among development, security, and operations teams for proactive threat mitigation.
- **Real-time monitoring and feedback loops**, allowing teams to assess and improve security measures based on actual performance.

Objective: To enable fast, secure software delivery by making security an integral part of the development pipeline.

Why DevSecOps Matters



- **Increasing Cyber Threats:** As cyber threats grow more sophisticated, integrating security early is essential to protect data and systems.
- **Faster Development Cycles:** Traditional security practices can slow down the pace; DevSecOps enables rapid, secure releases.
- **Cost Efficiency:** Catching vulnerabilities early reduces the cost of fixing issues post-deployment.
- **Compliance & Trust:** Automated checks ensure that applications meet regulatory standards, increasing reliability and trust.

Benefits of DevSecOps



- **Improved Security Posture:** Continuous integration of security practices reduces risks and improves resilience.
- **Higher Quality Software:** Early detection of issues leads to more stable, reliable software.
- **Faster Time-to-Market:** Automating security allows for rapid deployment without compromising safety.
- **Enhanced Team Efficiency:** Cross-functional collaboration minimizes bottlenecks and empowers teams to work together on security goals.

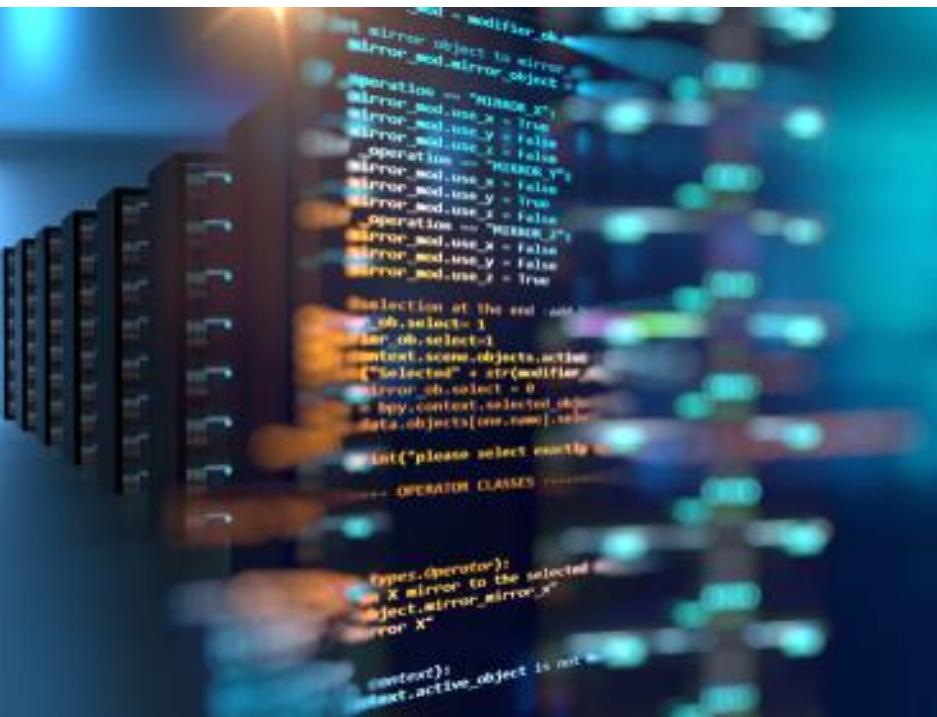
Infrastructure as Code



What is IaC?

Infrastructure as code (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files. Used with bare-metal as well as virtual machines and many other resources. Normally a declarative approach

Infrastructure as Code



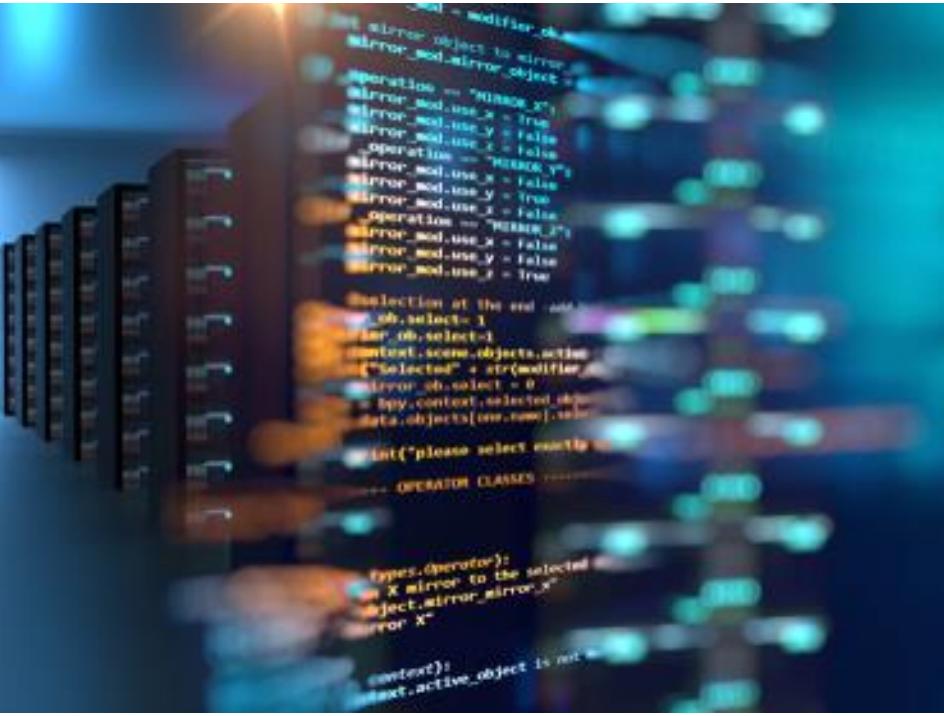
- Programmatically provision and configure components
- Treat like any other code base
 - Version control
 - Automated testing
 - data backup

Infrastructure as Code



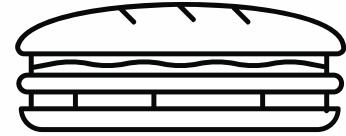
Provisioning infrastructure through software to achieve consistent and predictable environments.

Core Concepts



- Defined in code
- Stored in source control
- Declarative or imperative

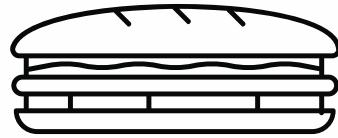
Imperative



```
# Make a sandwich  
get bread  
get mayo  
get turkey  
get lettuce
```

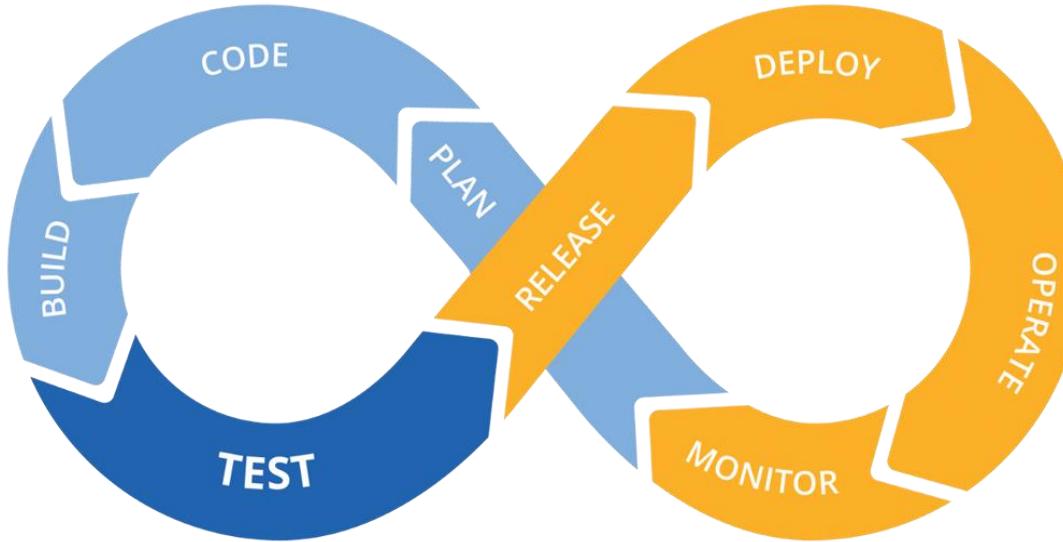
```
spread mayo on bread  
put lettuce in between bread  
put turkey in between bread on  
top of turkey
```

Declarative



```
# Make a sandwich
food sandwich "turkey" {
    ingredients = [
        "bread", "turkey", "mayo", "lettuce"
    ]
}
```

Security Integration in the CI/CD Pipeline



Continuously embedding and assessing security checks throughout the CI/CD pipeline to uphold robust, adaptable safeguards and maintain consistent compliance from development to deployment.

Introduction to Security in CI/CD



Integrating Security Throughout: Embedding security at every stage of the CI/CD pipeline enables teams to catch and address vulnerabilities as they arise, preventing issues from reaching production.

Ensuring Compliance and Readiness: By weaving security checks into the pipeline, code remains secure, compliant, and ready for production with each deployment.

Why Integrate Security in CI/CD?



Proactive Threat Mitigation: Identifies vulnerabilities as soon as code is committed.

Cost-Effective: Fixing issues early reduces the cost and complexity of remediation.

Continuous Protection: Ensures secure code delivery by integrating automated security checks.

Key Security Stages in the CI/CD Pipeline



1. **Code Scanning:** Analyzes source code for vulnerabilities.
2. **Dependency Checks:** Validates that third-party libraries are free from known vulnerabilities.
3. **Container Security:** Scans container images for security risks.
4. **Infrastructure as Code (IaC) Scanning:** Checks for misconfigurations in IaC templates.

Benefits of Security Integration in CI/CD



Early Detection: Vulnerabilities are identified before reaching production.

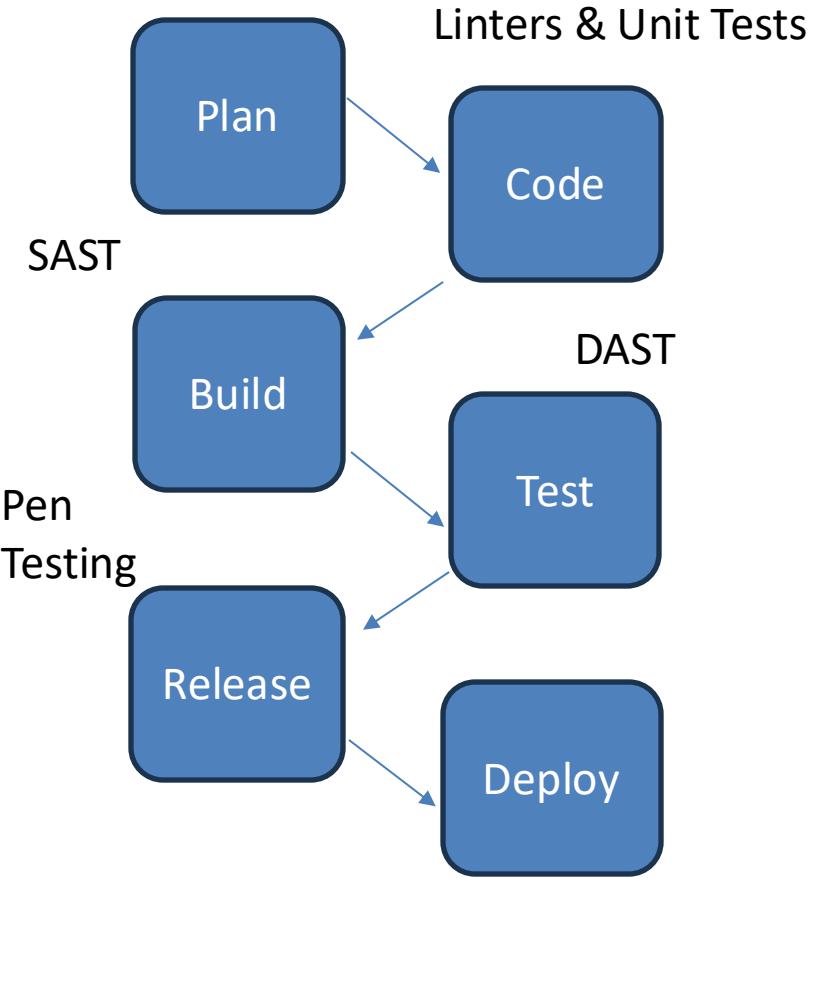
Automated Compliance: Security policies and compliance are enforced automatically.

Enhanced Team Efficiency: Developers receive instant feedback, allowing faster and safer code iterations.

Reduced Risk: Minimizes exposure to potential threats by ensuring secure code deployment.

Overview of CI/CD Security Integrations

Security Analysis



Code Scanning: Identifies vulnerabilities directly in the source code before merging.

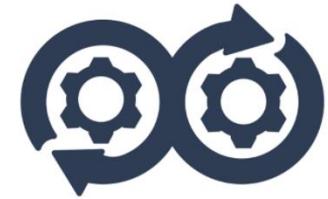
Dependency Scanning: Checks third-party libraries for known vulnerabilities to ensure secure dependencies.

Container Security: Scans container images for configuration issues and vulnerabilities.

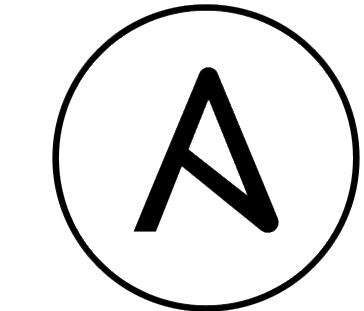
Infrastructure as Code (IaC) Scanning: Validates cloud and infrastructure templates to catch misconfigurations early.

Dynamic Application Security Testing (DAST): Tests the running application for security weaknesses from an attacker's perspective.

Case Study: Media Corporation

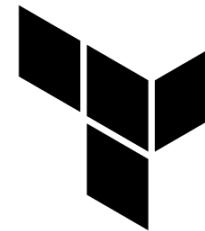


Git



ANSIBLE

git



HashiCorp
Terraform

 python™

Git

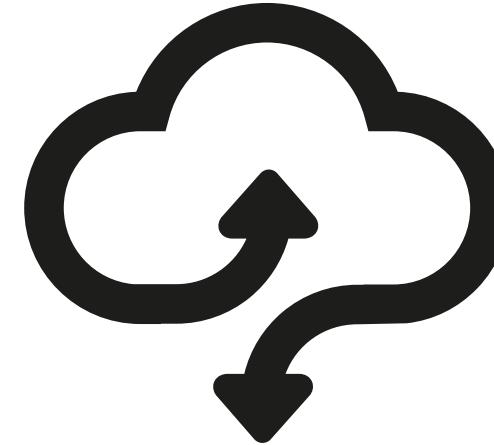


git



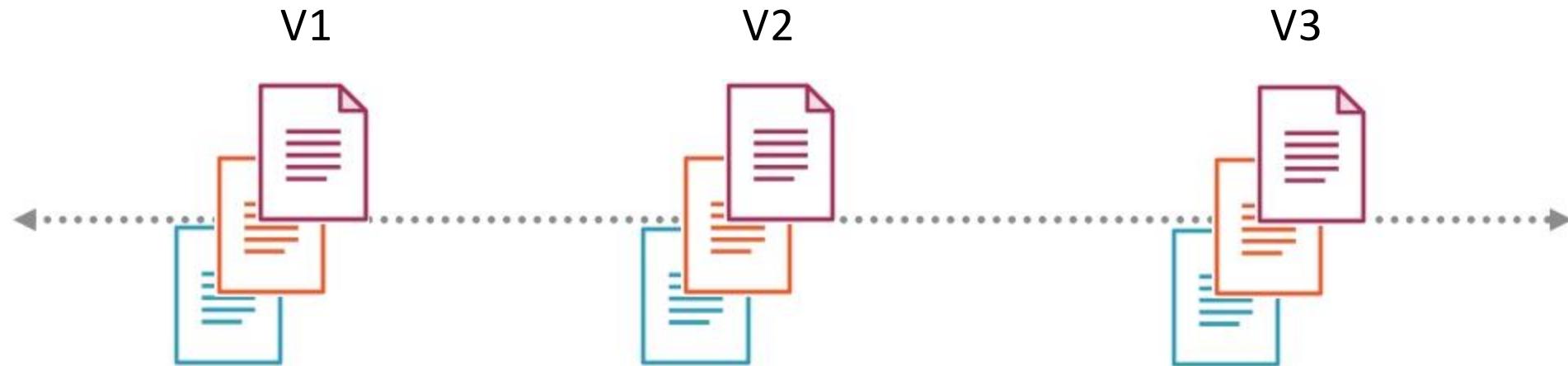
Git

git



What is Git?

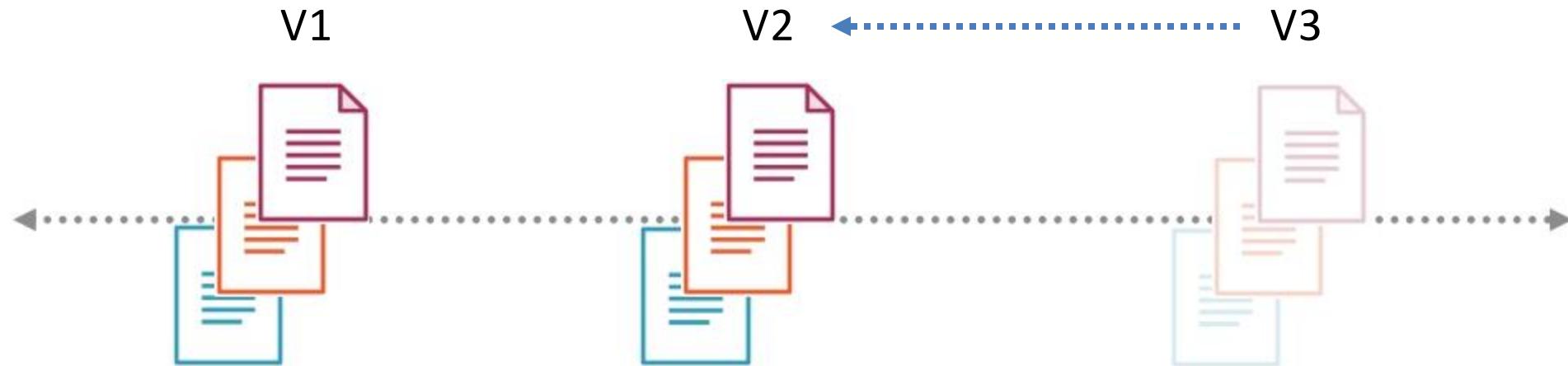
Version Control System



What is Git?

Version Control System

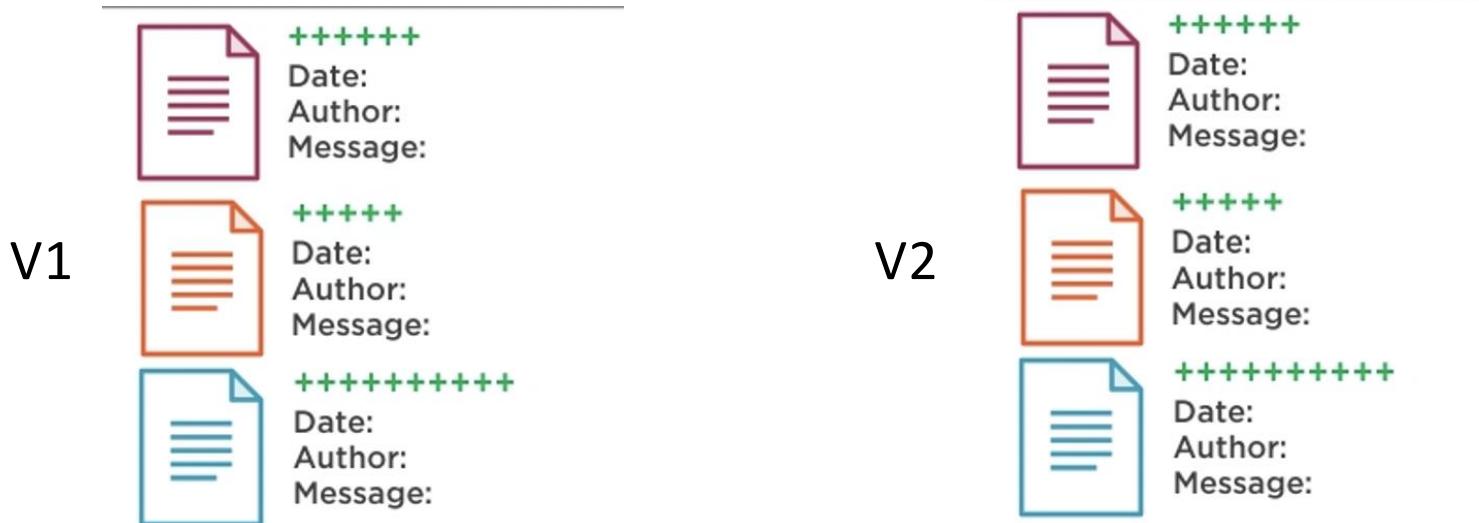
- Software designed to record changes to files made over time
- Ability to revert back to a previous file version or project version



What is Git?

Version Control System

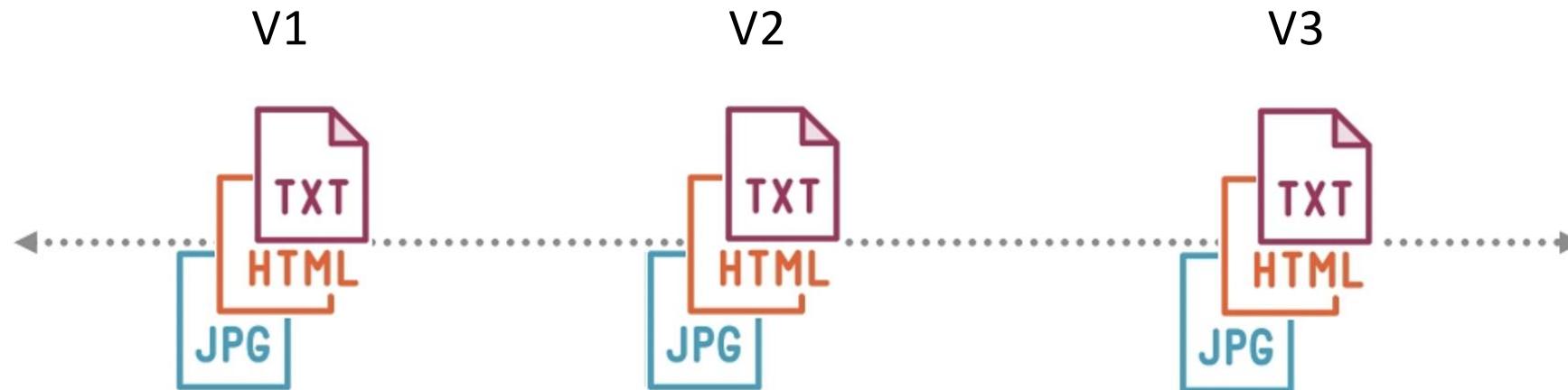
- Software designed to record changes to files made over time
- Ability to revert back to a previous file version or project version
- Compare changes made to files from one version to another



What is Git?

Version Control System

- Software designed to record changes to files made over time.
- Ability to revert back to a previous file version or project version.
- Compare changes made to files from one version to another.
- Ability to version control any plain text file, not just source code.



Git use-case



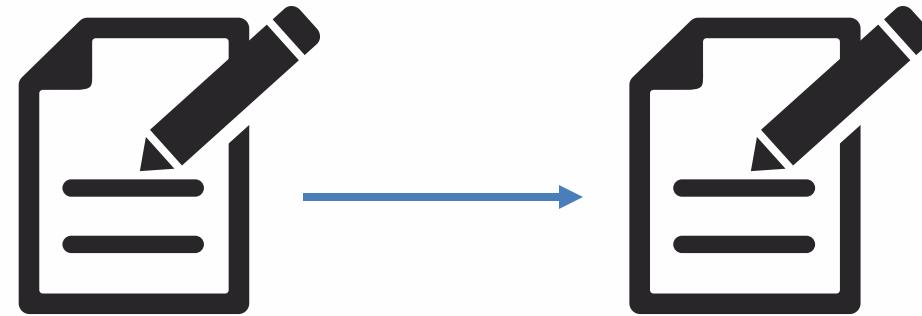
Git use-case



Latte



Git use-case



POP QUIZ: Dangers of copying files



What are some dangers of Steve's current process?

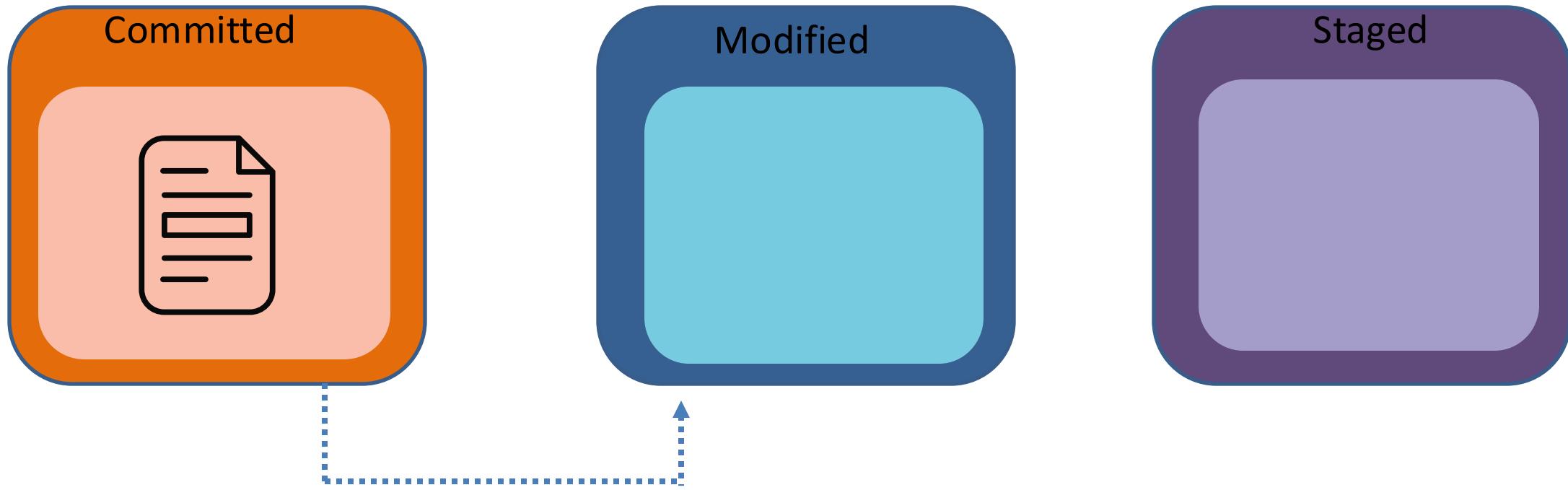
POP QUIZ: Dangers of copying files



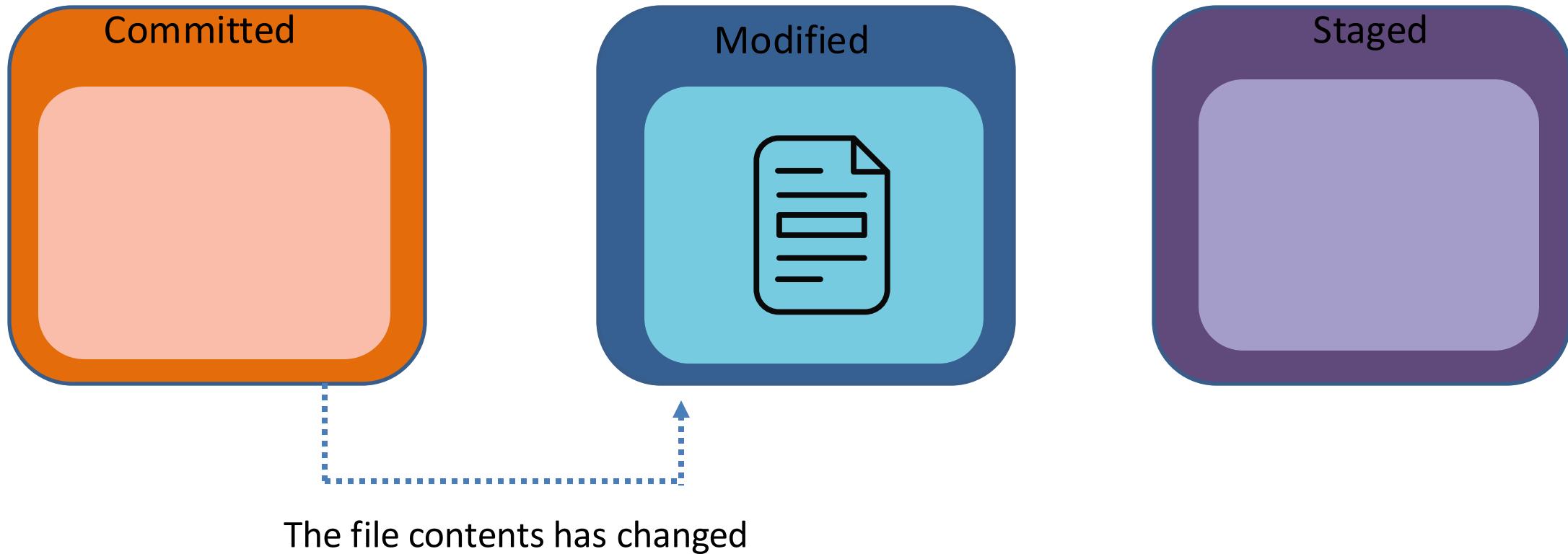
What are some dangers of Steve's current process?

- Files live in one location
- Accidentally overwrite existing recipes

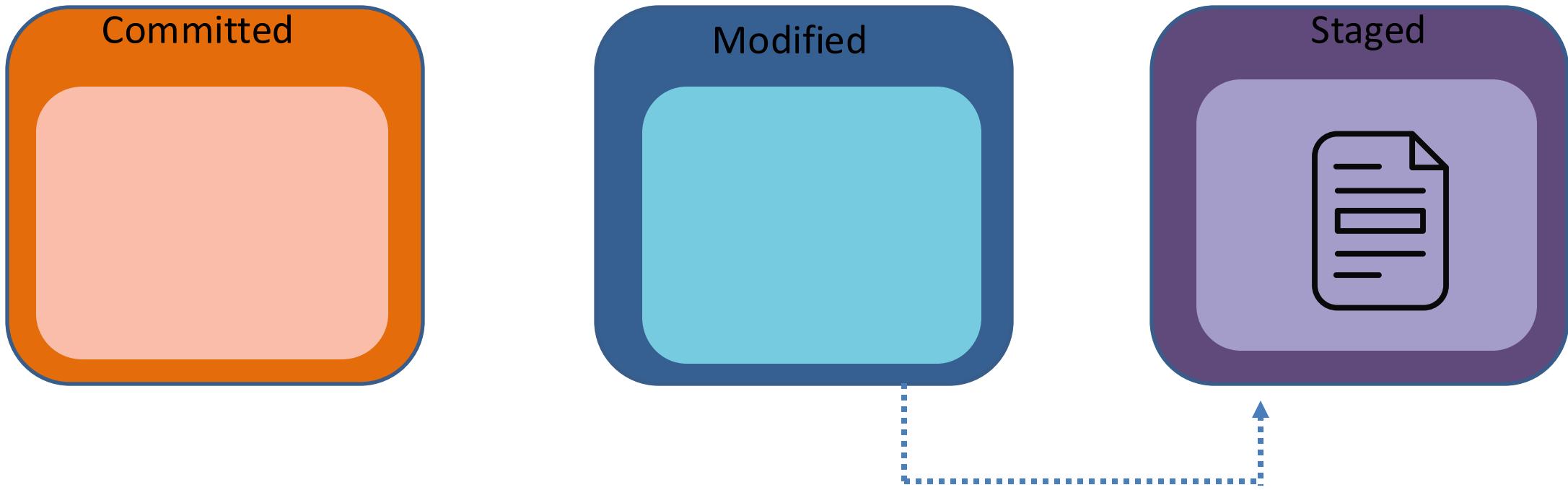
Three stages of a file



Three stages of a file

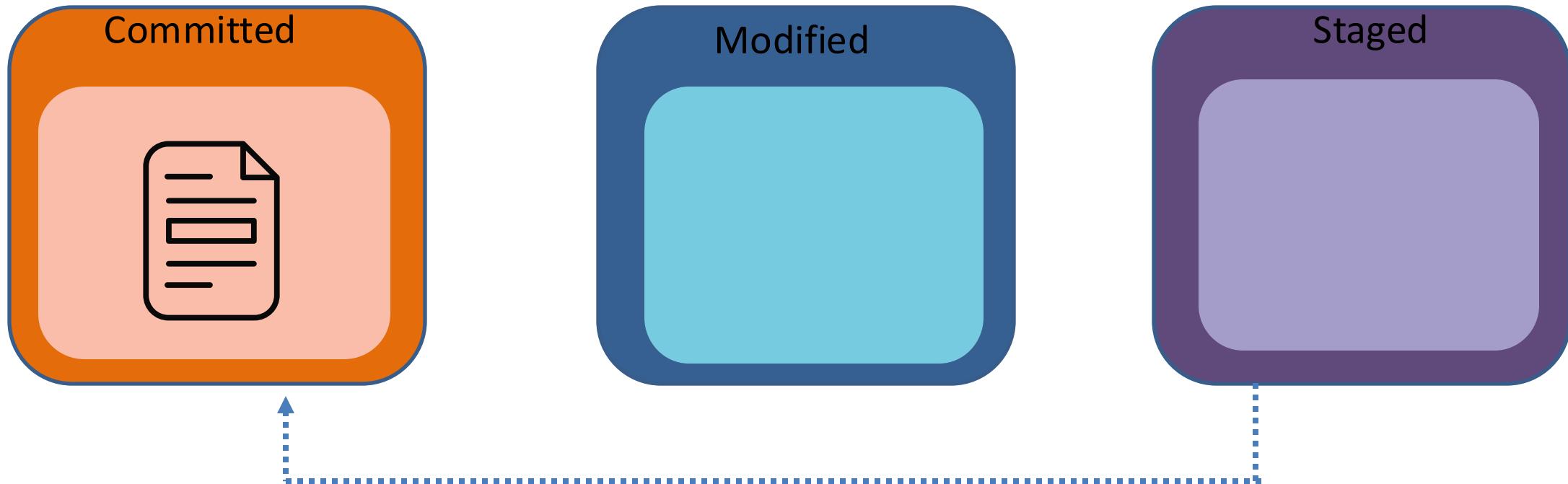


Three stages of a file



Changes are marked for commit

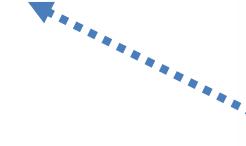
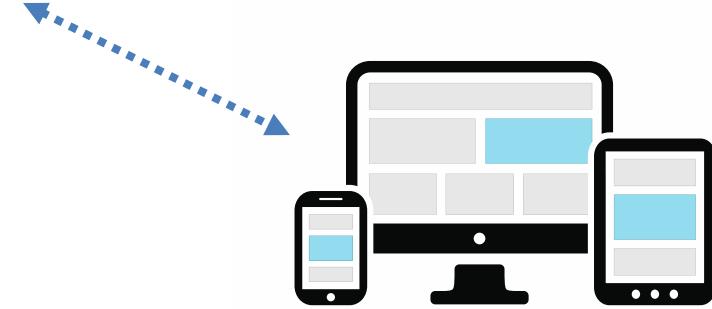
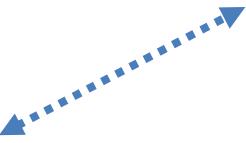
Three stages of a file



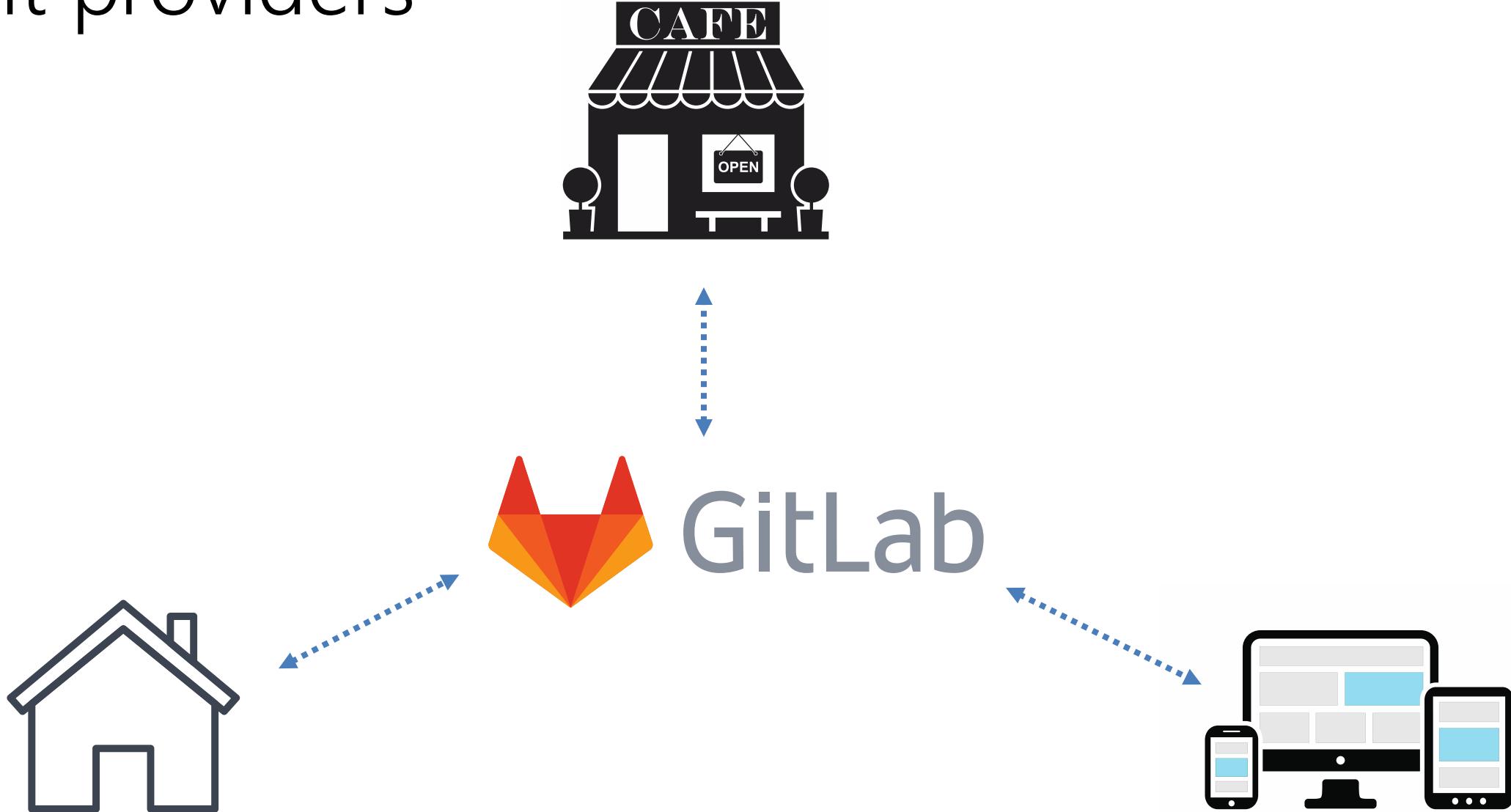
Git providers



GitHub



Git providers



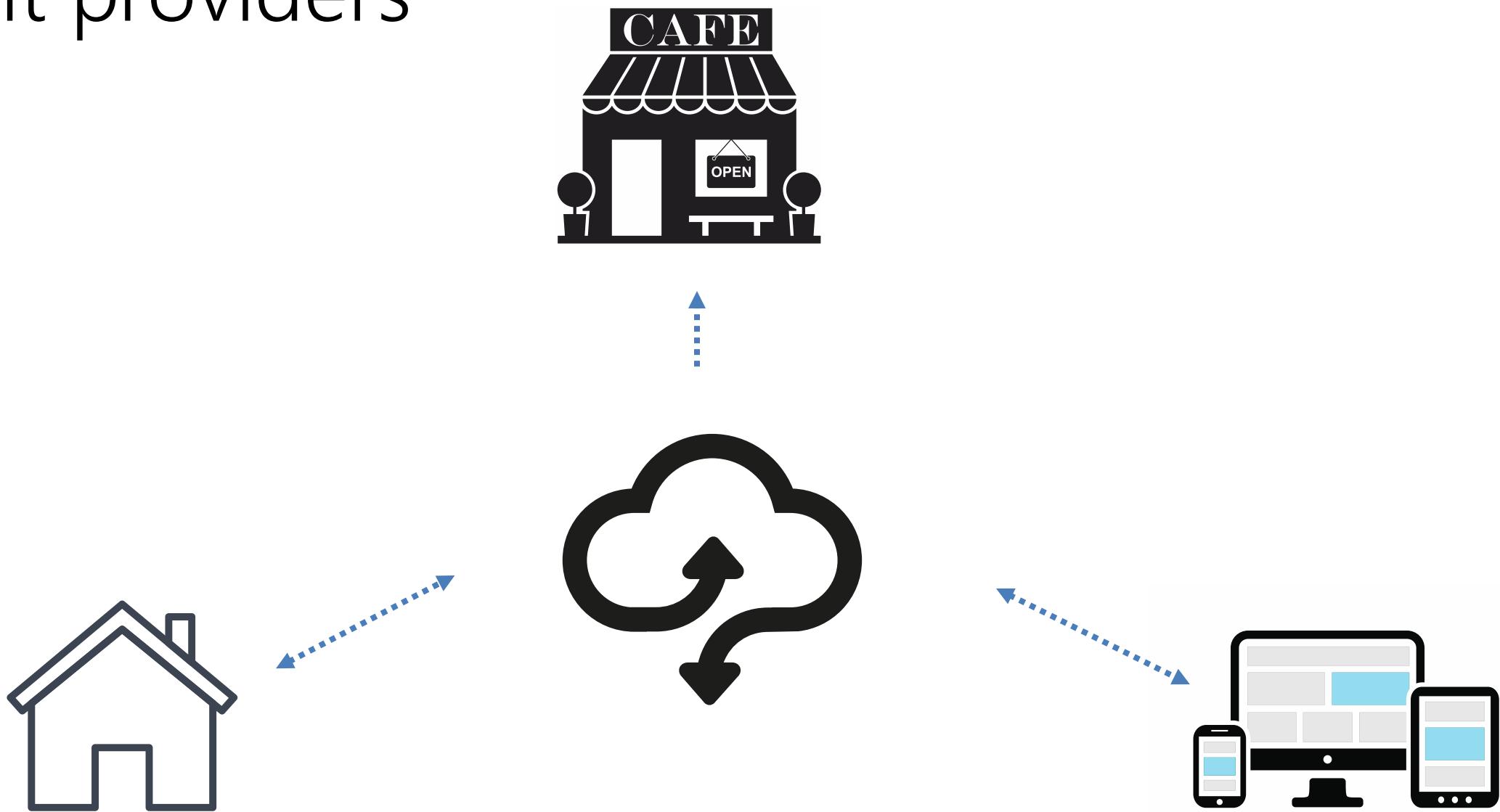
Git providers



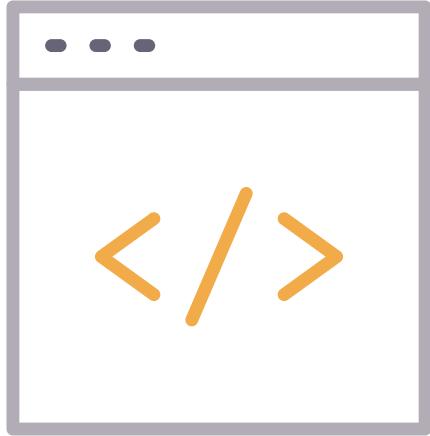
Bitbucket



Git providers



Git basic commands

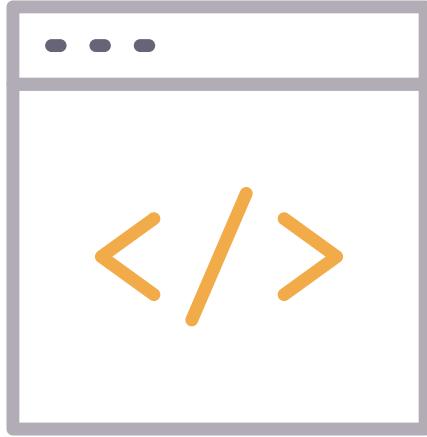


Clone from upstream:

```
$ git clone git://git.kernel.org/pub/scm/.../linux.git my-linux  
$ cd my-linux
```

- **git clone** is used to copy a repository
 - You can also clone a local repository
 - `git clone /path/to/repository`

Git basic commands



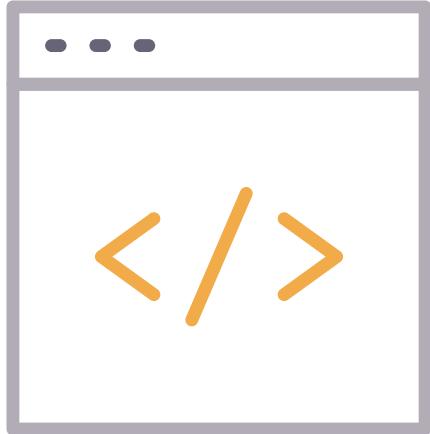
Start a new Git repository for an existing code base

```
$ cd /path/to/my/codebase  
$ git init      (1)  
$ git add .     (2)  
$ git commit    (3)
```

1. Create a `/path/to/my/codebase/.git` directory.
2. Add all existing files to the index.
3. Record the pristine state as the first commit in the history.

- Use `git init` to create a new local repository.
- Alternatively, you can create a repository within a new directory by specifying the project name: `git init [project name]`

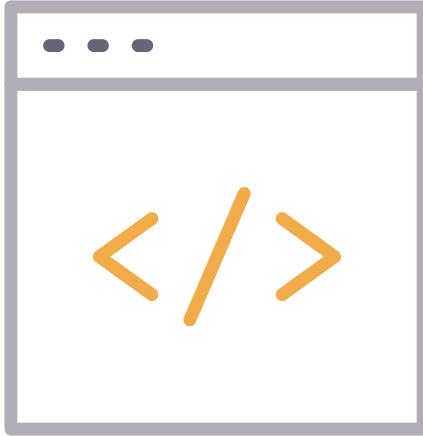
Git basic commands



```
$ git add git-*.*sh
```

- Use `git add <files/directories>` to add files to staging area.

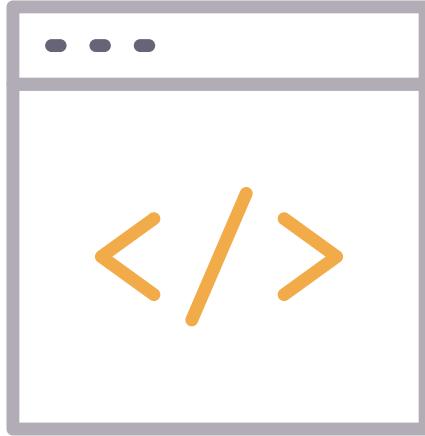
Git basic commands



```
$ git commit
```

- `git commit` will create a snapshot of the changes and save it to the git directory.
- `-m` will pass a commit message: `git commit -m "commit message"`

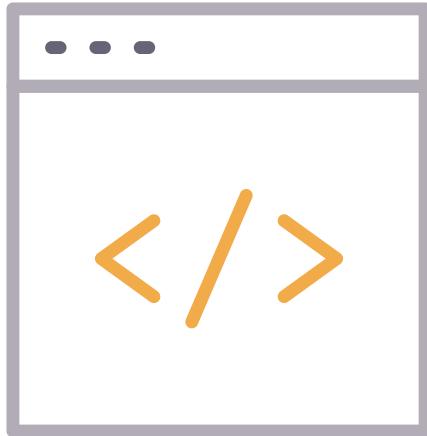
Git basic commands



```
git config --add core.gitproxy '"proxy-command" for example.com'
```

- git config can be used to set user-specific configuration values like email, username, file format, and so on.
- git config --global user.email youremail@example.com
- --global sets value for all repositories

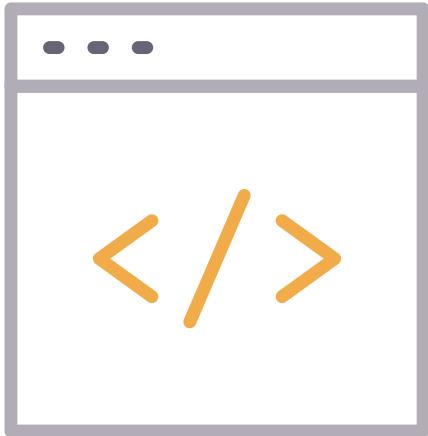
Git basic commands



```
git status [<options>...] [--] [<pathspec>...]
```

- `git status` displays the list of changed files together with the files that are yet to be staged or committed.

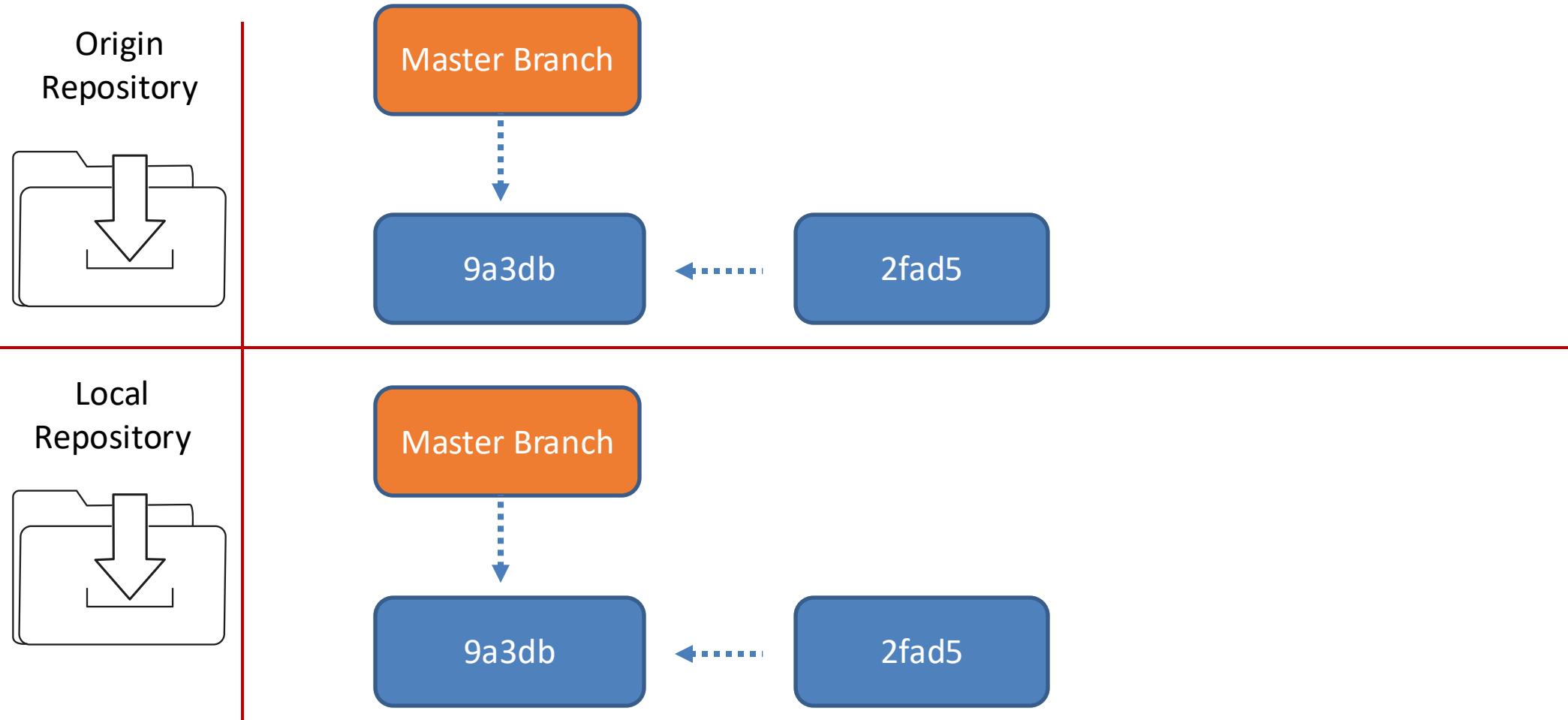
Git basic commands



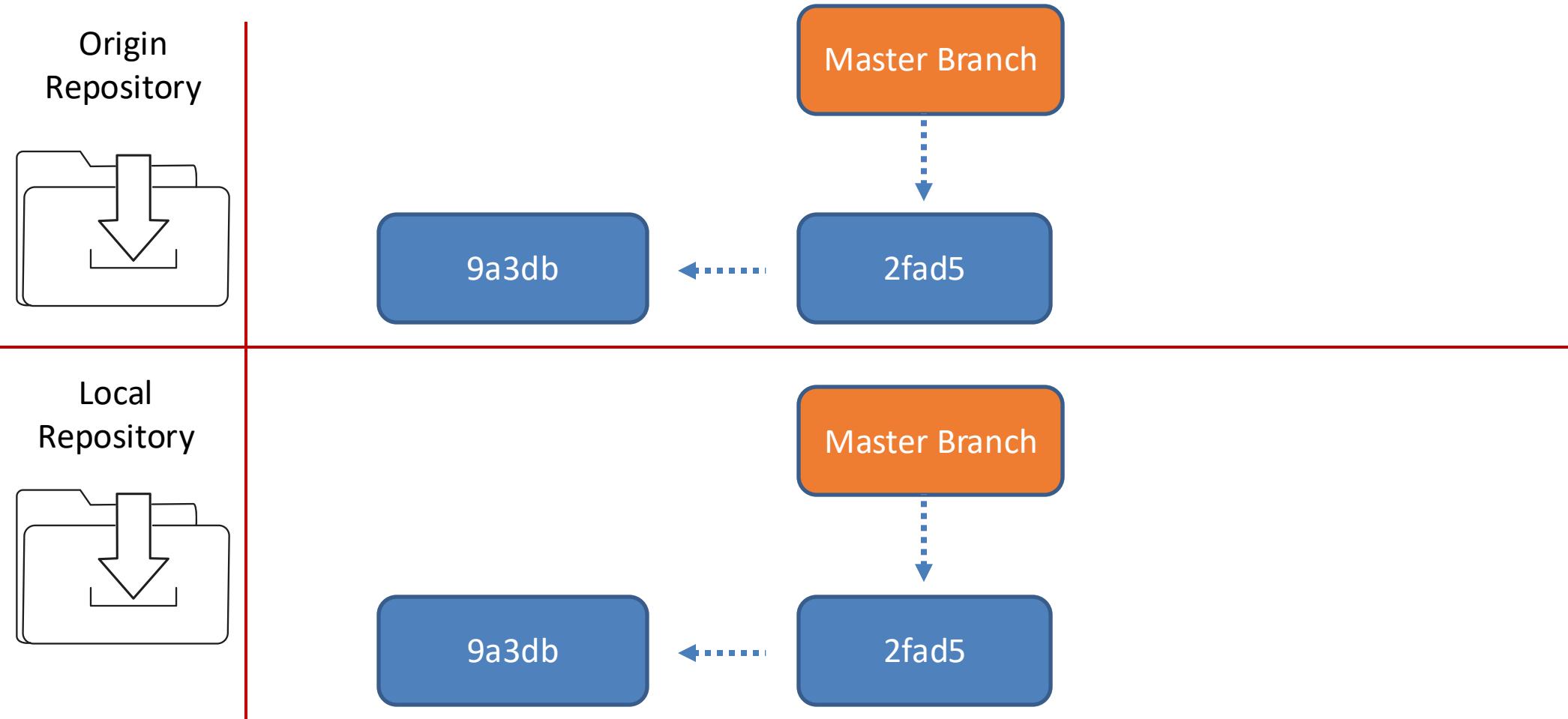
```
git push origin master
```

- git push **is used to send local commits to the master branch of the remote repository.**
- git push origin <branch>

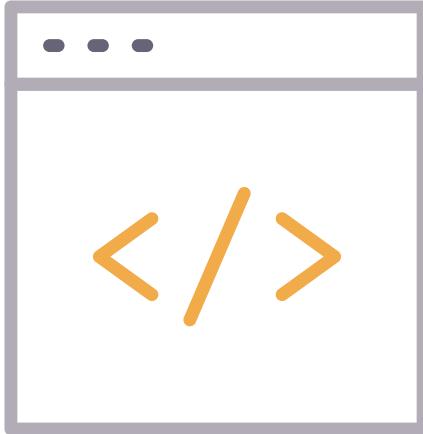
Git commit process



Git commit process



Git diff

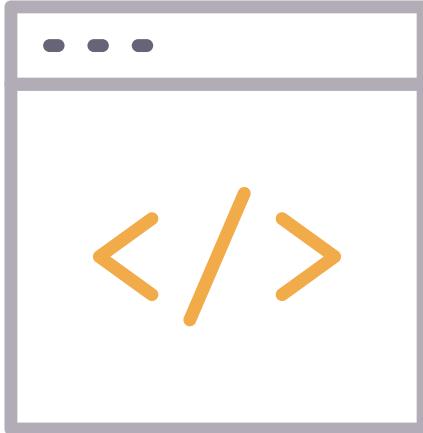


Various ways to check your working tree

```
$ git diff          (1)  
$ git diff --cached (2)  
$ git diff HEAD     (3)
```

- `git diff` lists down conflicts.
- Use it to show changes that have been made but have not been committed.

Git log

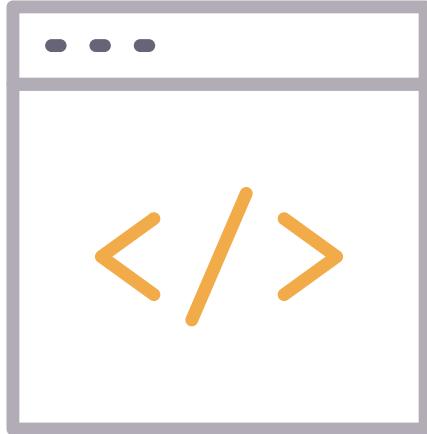


```
git log --no-merges  
Show the whole commit history, but skip any merges
```

```
commit a68112e12f896ca028f25a46e58aaaf46360cb847  
Author: Jason Smith <jason@smithss.org>  
Date:   Sun Jan 6 21:12:35 2019 +0000  
  
    initial
```

- `git log` is used to see the repositories history by listing certain commit details.

Git blame

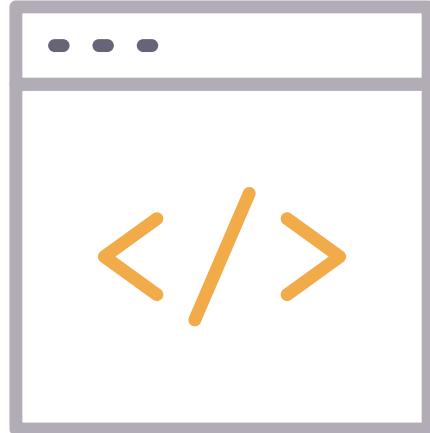


```
git blame CNAME
```

```
a76f1de5 (Jason Smith 2020-04-19 22:52:29 -0700 1) kickstart.innovationin.software  
5948ef48 (Jason Smith 2020-04-19 22:51:19 -0700 2)
```

- `git blame` is used to see which users changed a file.

Git blame



```
$ git commit ...
$ git reset --soft HEAD^      (1)
$ edit                      (2)
$ git commit -a -c ORIG_HEAD (3)
```

Undo commits permanently

```
$ git commit ...
$ git reset --hard HEAD~3    (1)
```

- `git reset` **undo changes**
- `--soft` `--mixed` or `--hard` depending on situation.
- `--soft` Does NOT change local working directory. Index still points to latest commit
- `--mixed` Does NOT change local working directory. Index points to reverted commit
- `--hard` Changes local working directory and index to point to reverted commit. Deletes all comments and files from original commit.

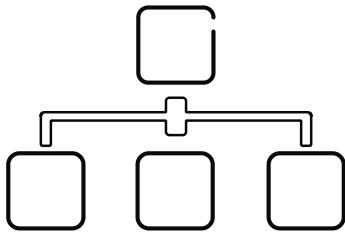
POP QUIZ: Your development flow



How does your team add new features to your applications?

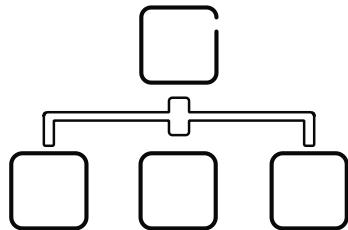
- Feature branches?
- All changes to master?

Software Development flows



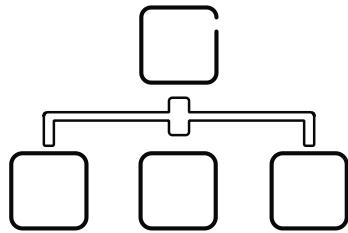
- Centralized flow
- GitHub
- Featured branch

Centralized flow

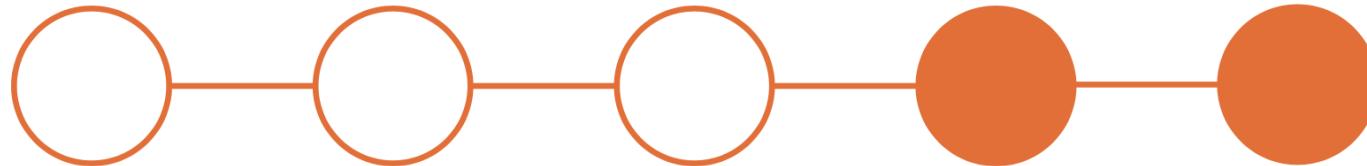


- Uses a centralized repository
- End users 'clone' repository
- Pull changes from central repository to local
- Make changes
- Push local changes back to centralized repository
- Similar to SVN

Centralized flow

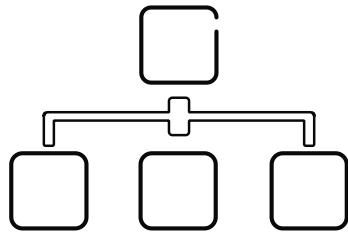


Central repository



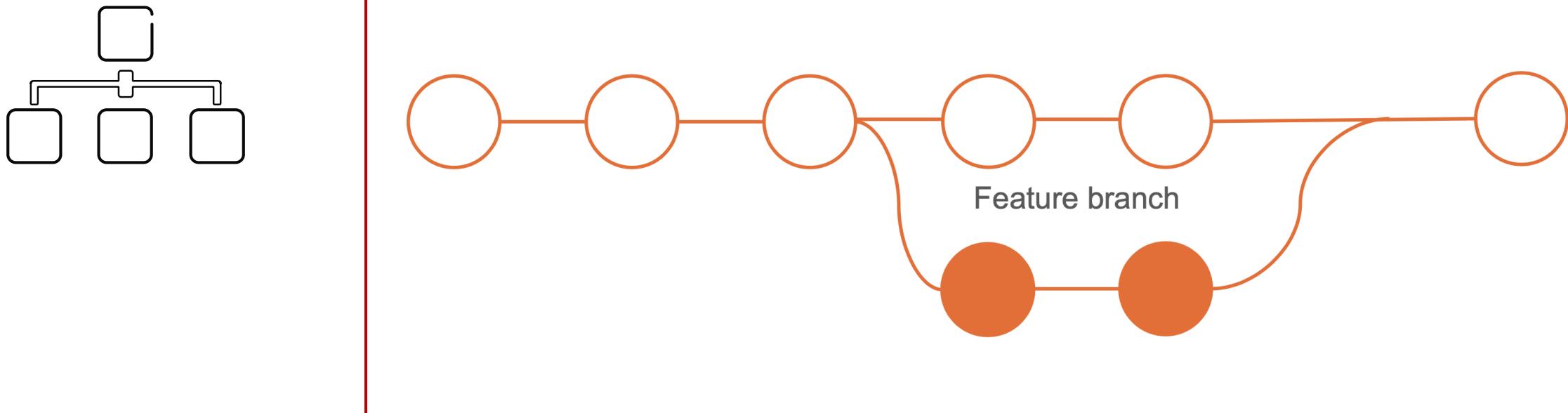
Local pushed changes

Feature branch flow

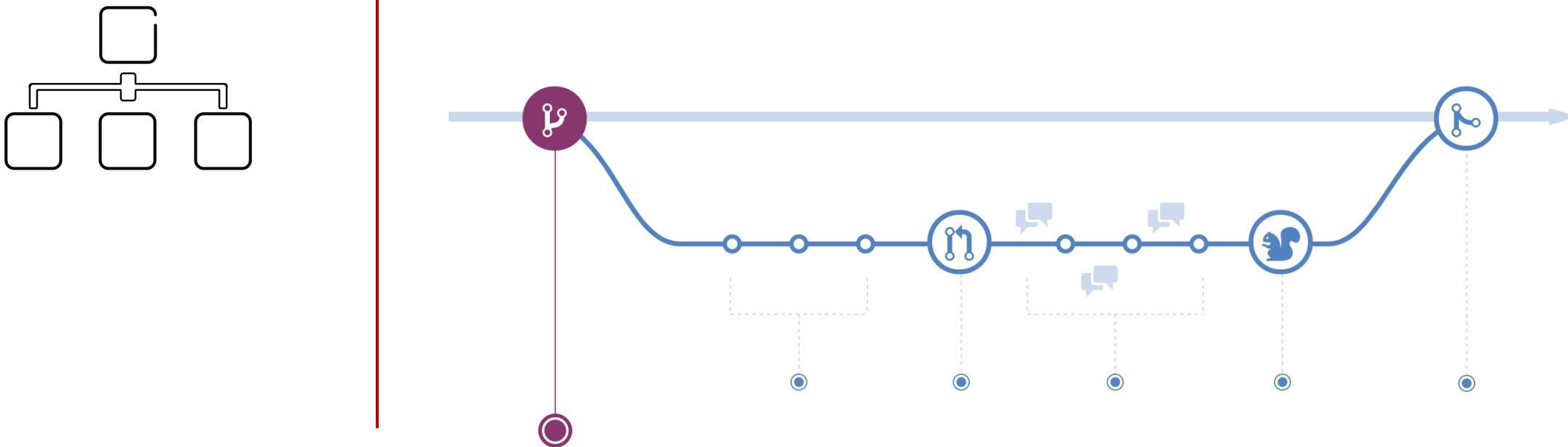


- Create a feature branch
- Add new code
- Open a pull request
- Merge code back into Master

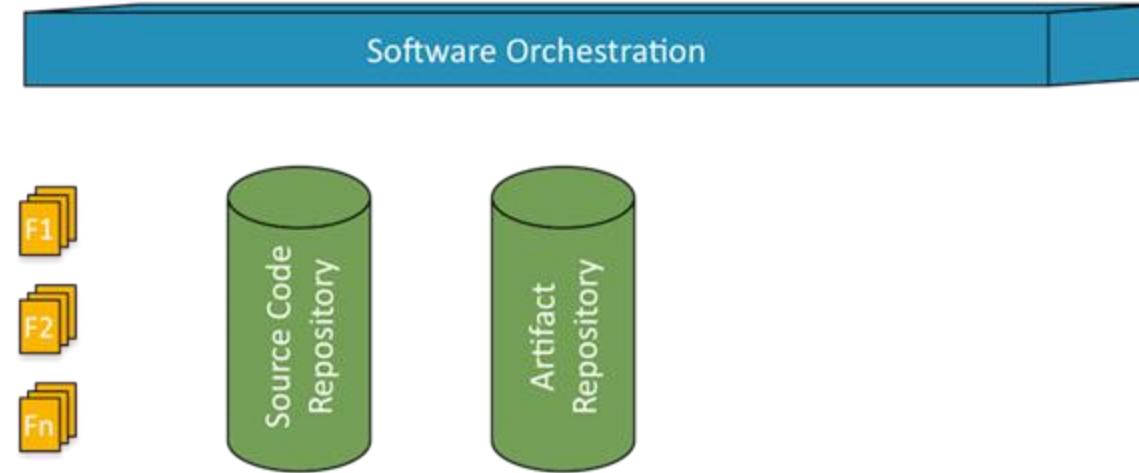
Feature branch flow



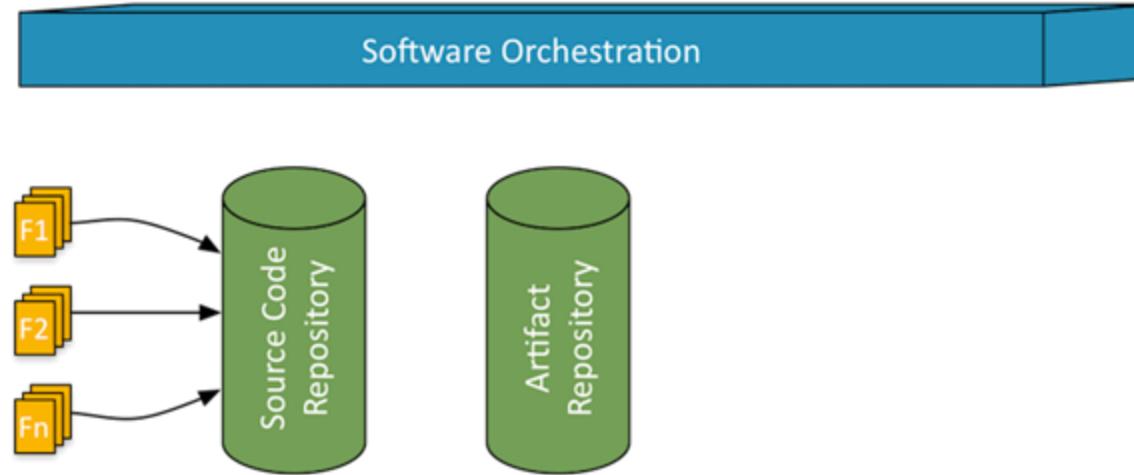
GitHub flow



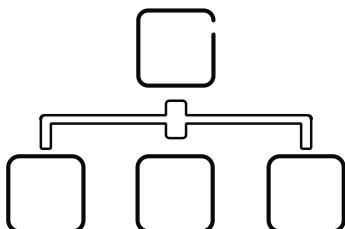
GitHub flow



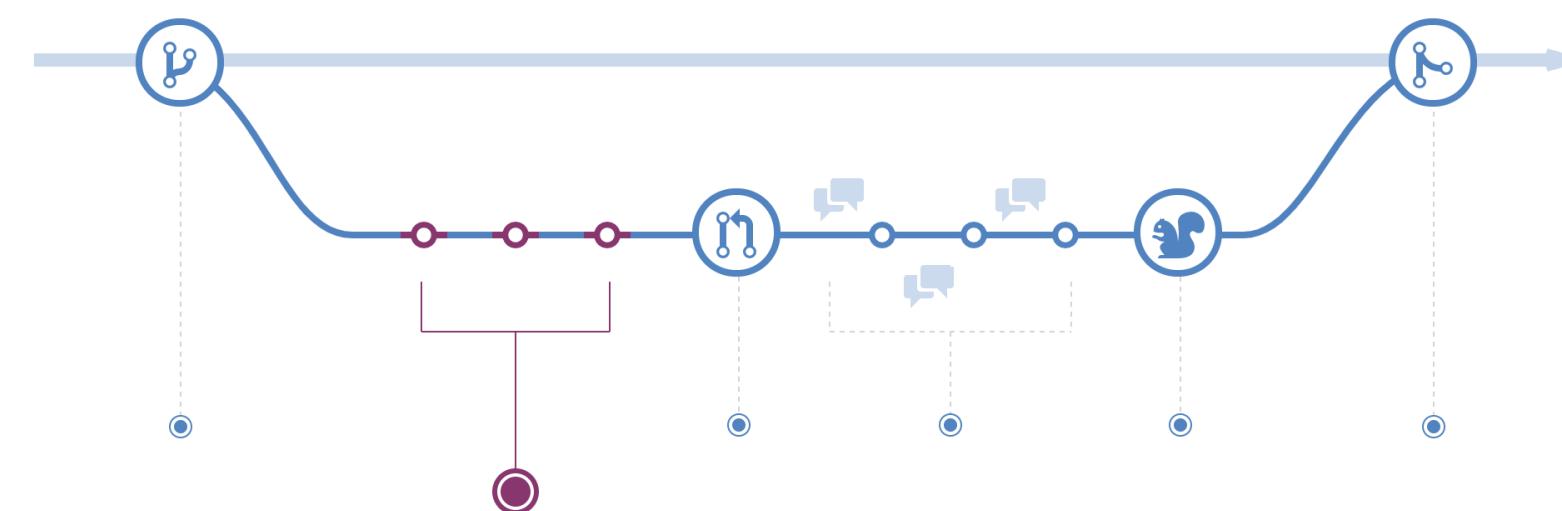
GitHub flow



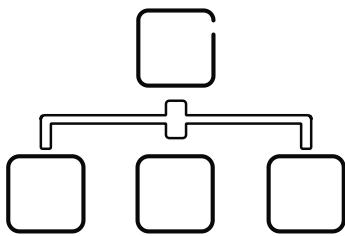
GitHub flow



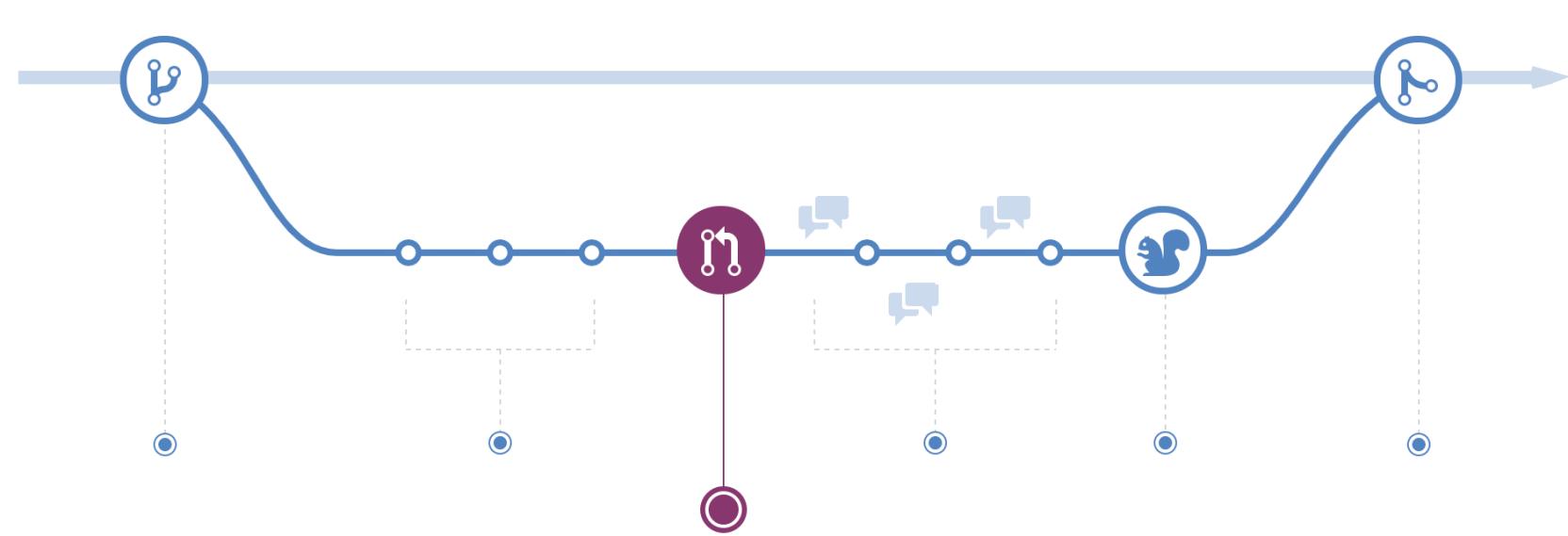
1. Create a branch
2. Checkout branch and make changes



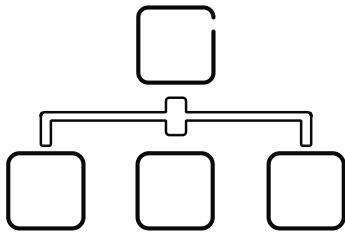
GitHub flow



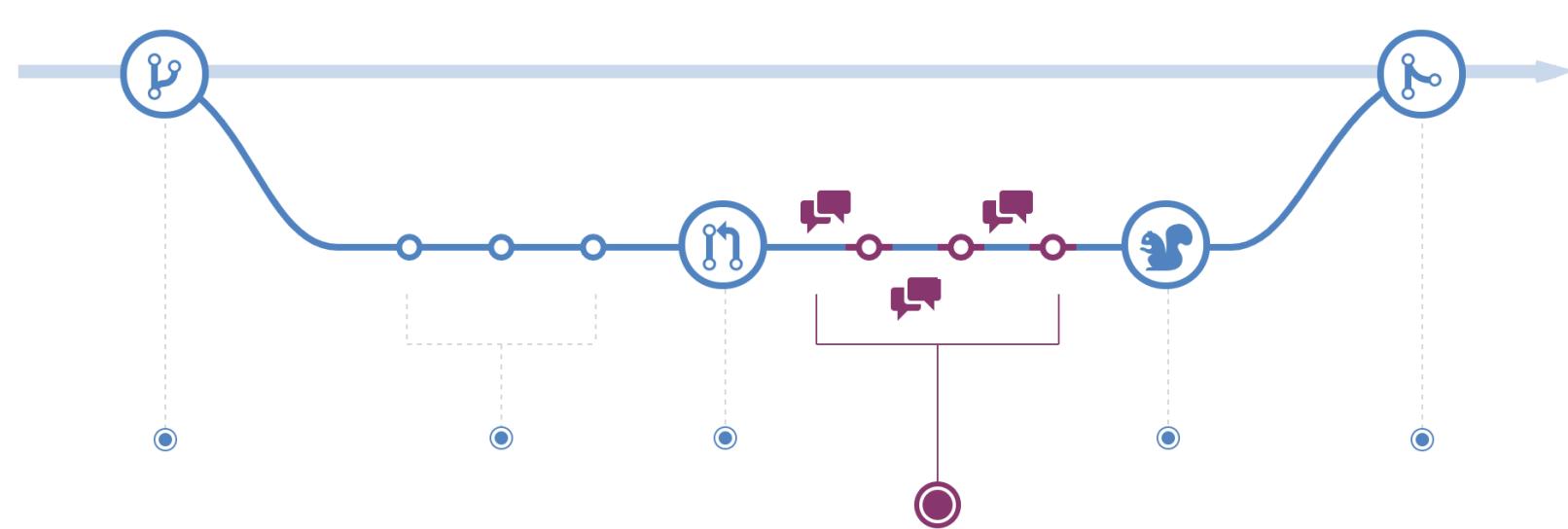
1. Create a branch
2. Checkout branch and make changes
3. Open pull request



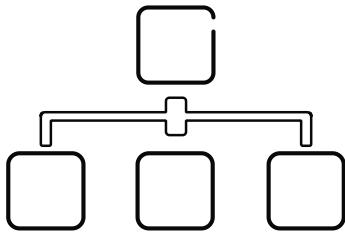
GitHub flow



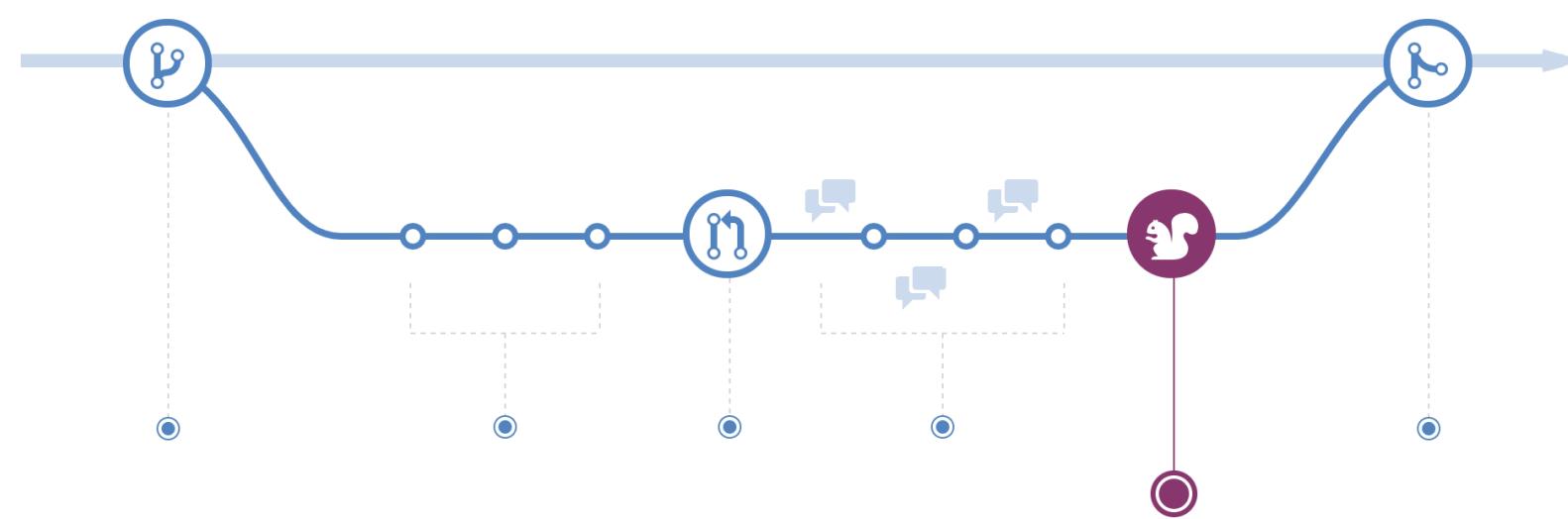
1. Create a branch
2. Checkout branch and make changes
3. Open pull request
4. Peer review



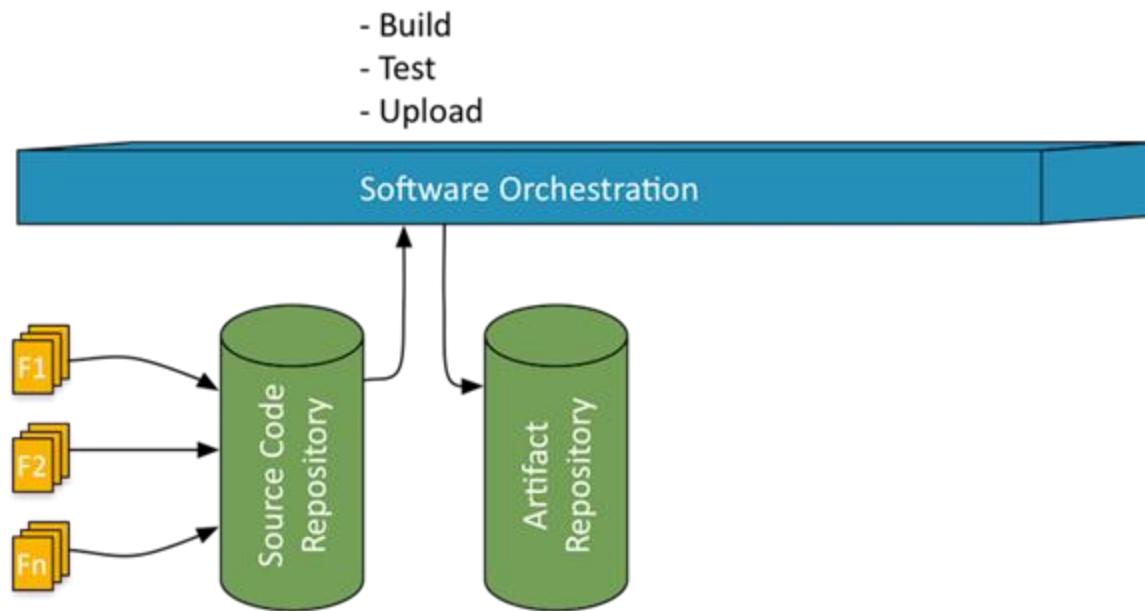
GitHub flow



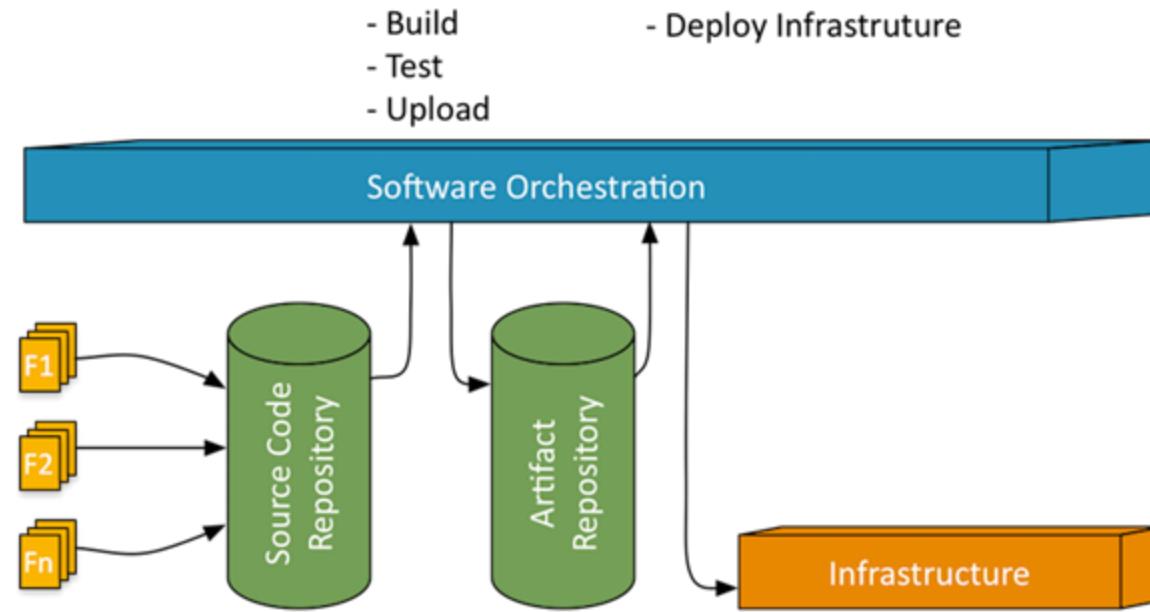
1. Create a branch
2. Checkout branch and make changes
3. Open pull request
4. Peer review
5. Deploy



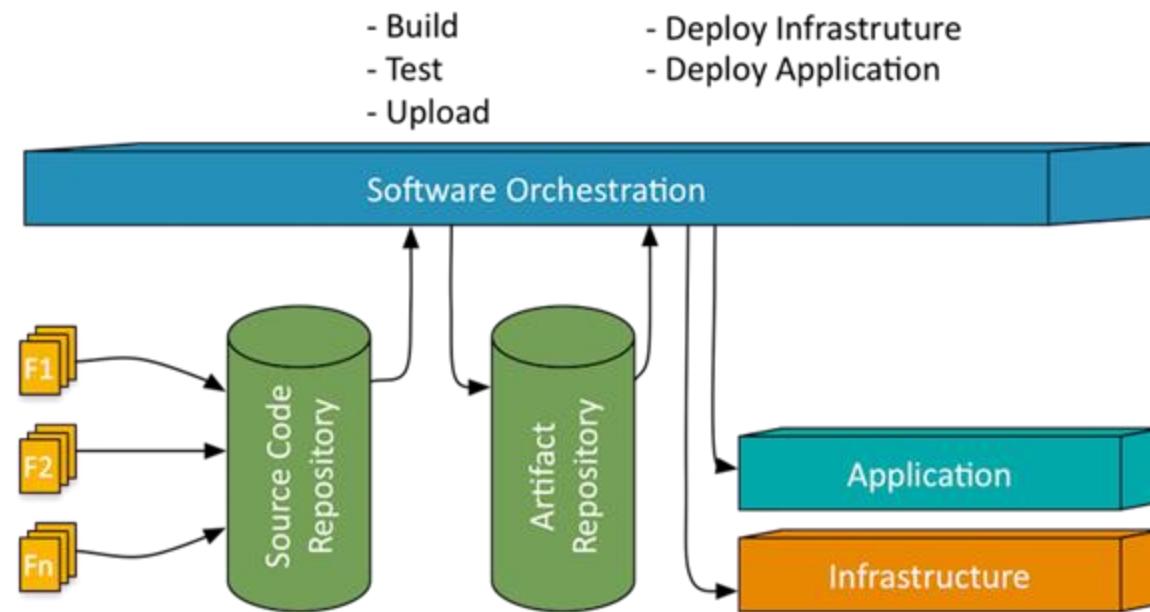
Continuous Integration & Testing



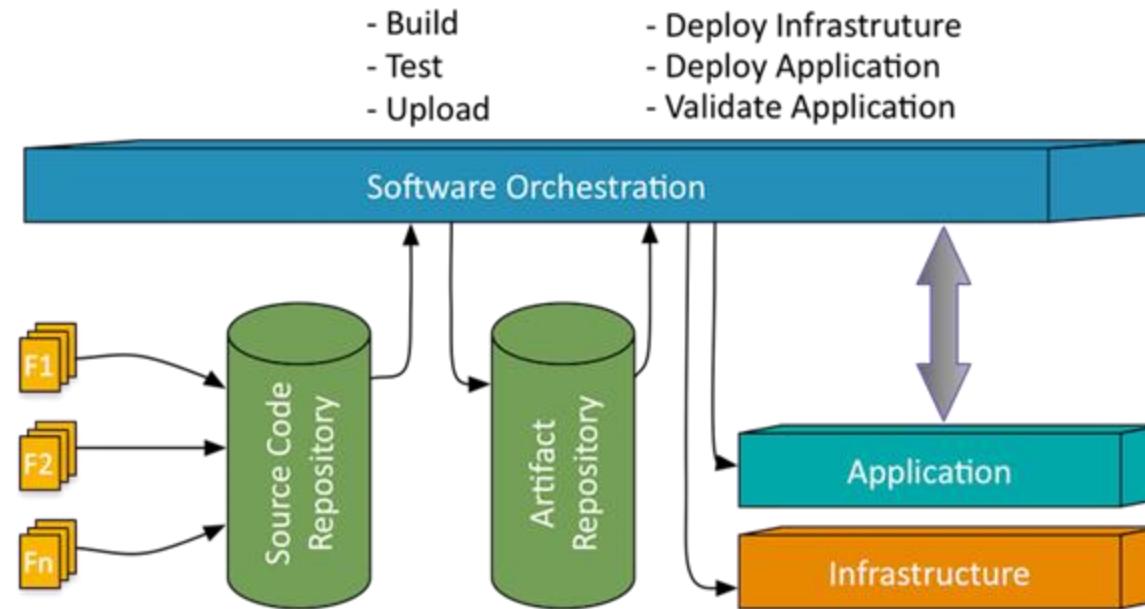
Continuous Release and Deployment



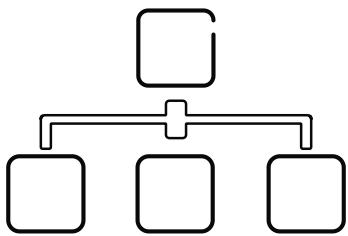
Continuous Release and Deployment



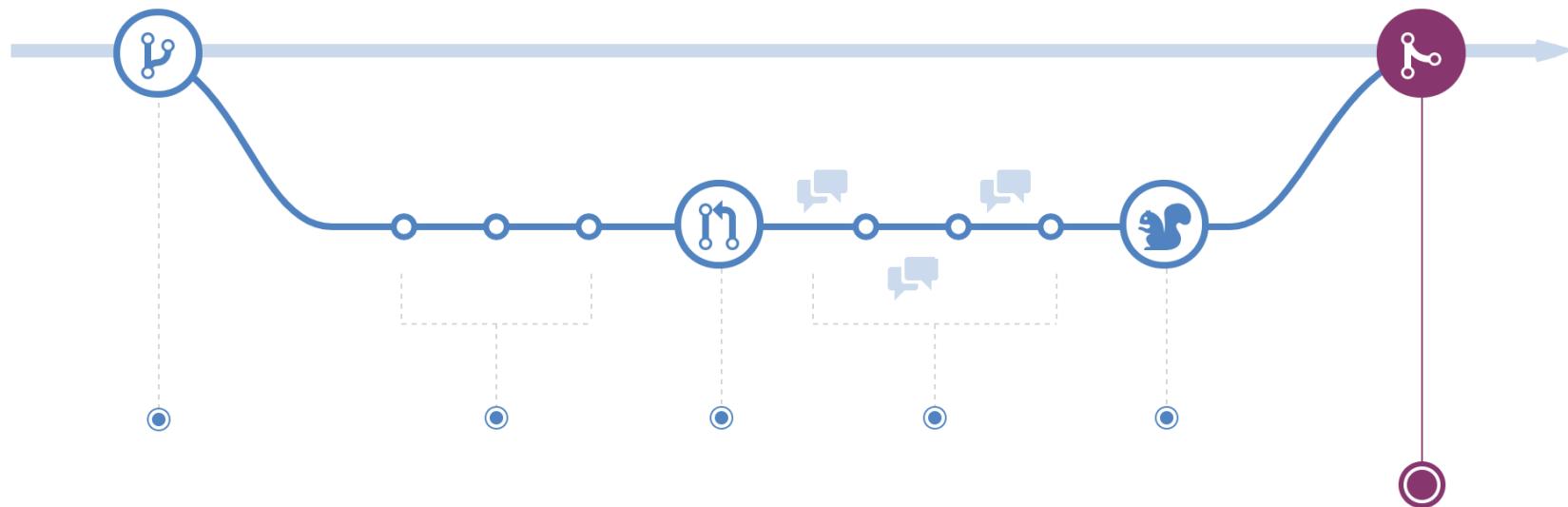
Continuous Release and Deployment



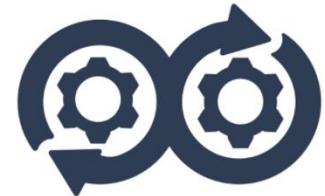
GitHub flow



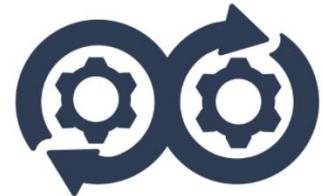
1. Create a branch
 2. Checkout branch and make changes
 3. Open pull request
 4. Peer review
 5. Deploy
 6. Merge into Master branch



Lab: Git



Configuration Management



Configuration Drift



POTHOLE

- Your infrastructure requirements change
- Configuration of a server falls out of policy
- Manage with Infrastructure as Code (IAC)
 - Terraform
 - Ansible
 - Chef
 - Puppet

Manual



POP QUIZ: Manual tasks



What manual tasks do you deal with repeatedly?

Can they be automated?

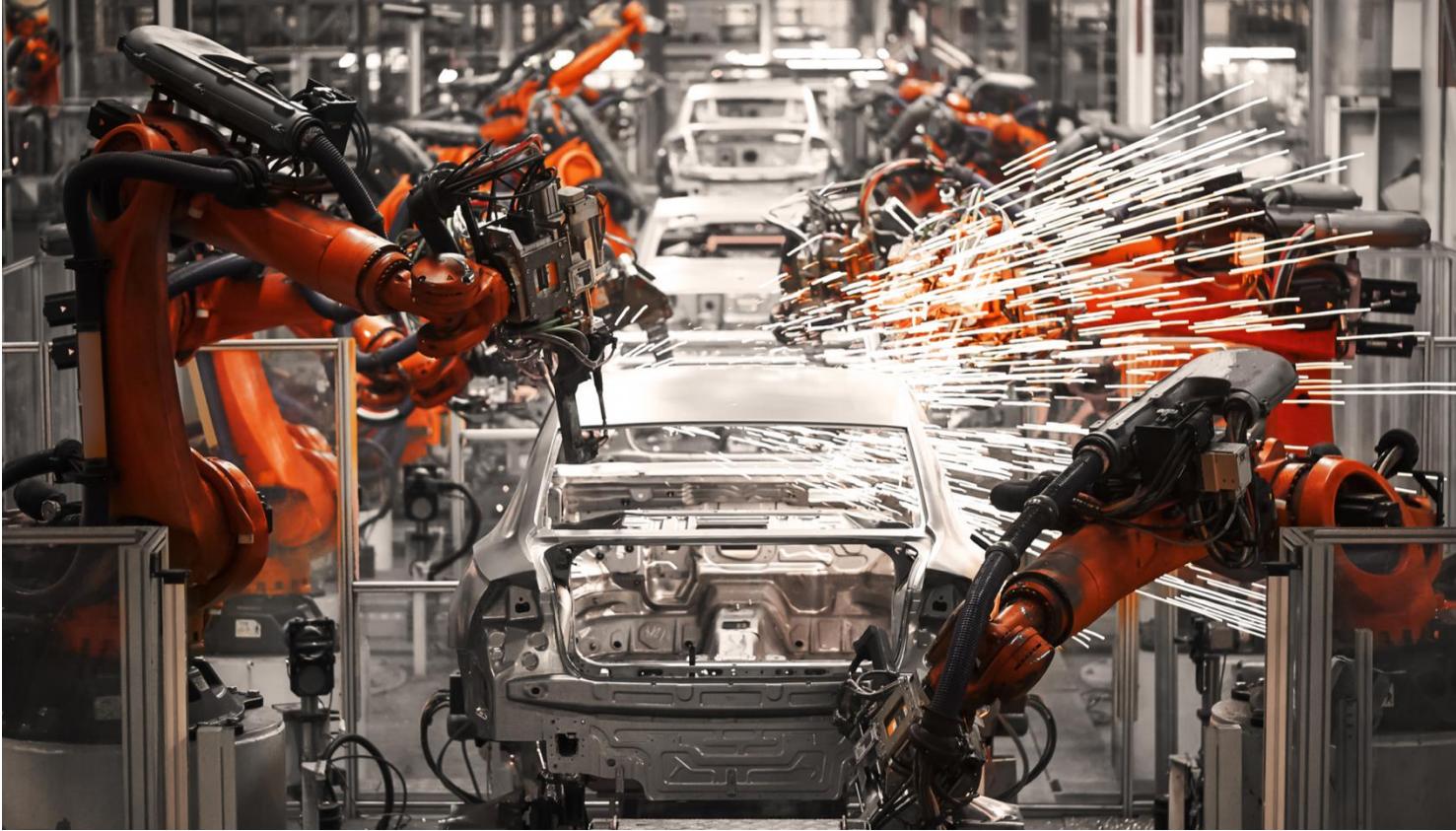
Common manual tasks



Installation and configuration:

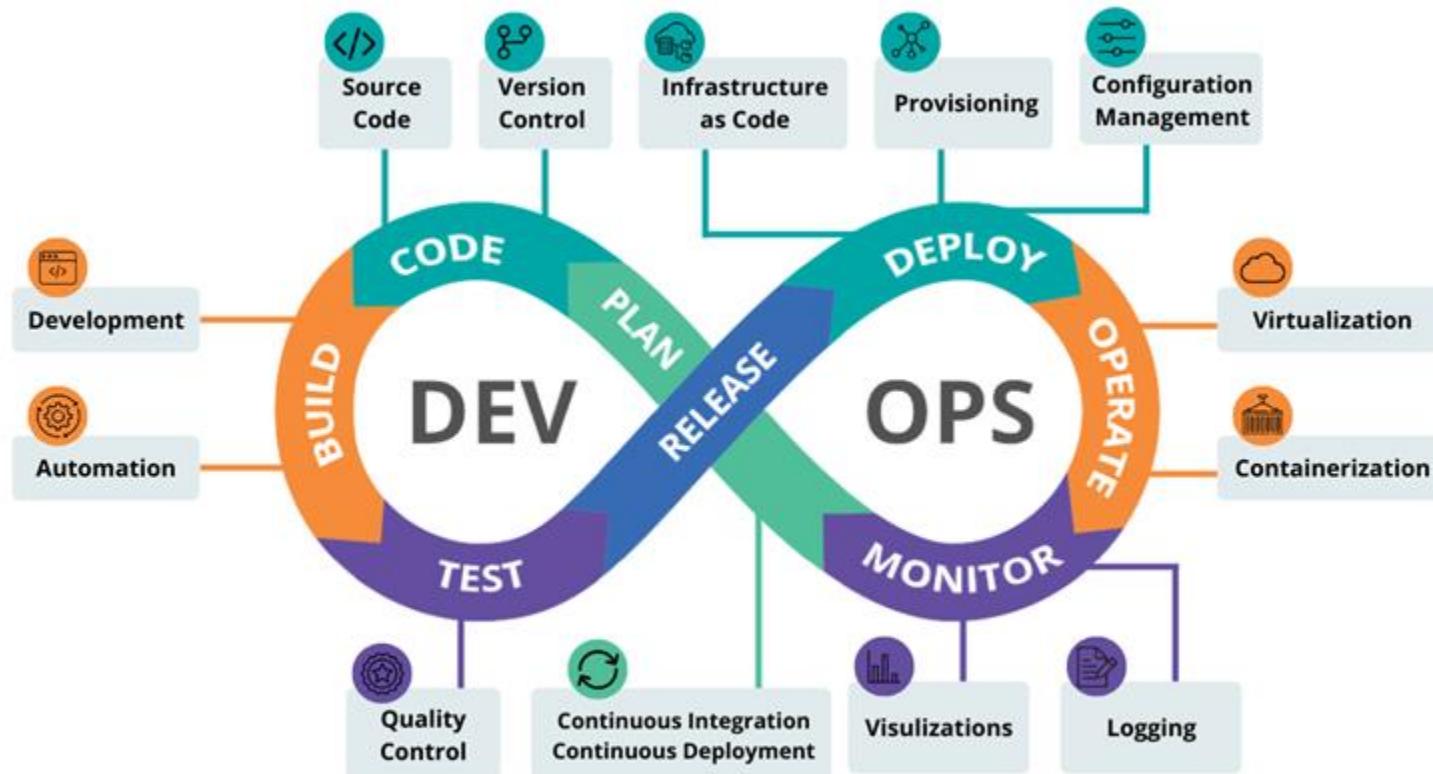
- Operating System
 - Runtime
 - Libraries
 - Utilities
 - Configuration files
-
- Build application
 - Test application
 - Deploy application

Automated



What is DevOps?

DevOps is the process of integrating Developer and Operation teams in order to improve collaborations and productivity



Components/Services of Azure DevOps



Azure Boards

Allows Work item tracking, Agile planning, Power BI visualization, and similar other reporting tools.



Azure Test Plans

Provides integrated planning and investigation of testing solutions.



Azure Repos

Provides full-support for cloud-hosted private repositories.



Azure Artifacts

Package management
Support for Maven, npm, NuGet and Python package feeds from private or public sources.



Azure Pipelines

Defines CI/CD- Continuous Integration and Continuous deployment process with support for containers and Kubernetes.



Overview | Summary

The summary page holds information about the project and is the Home Page of your project. Here you can add a basic description, tags to your project and also show a Readme file or a Wiki page. Not only that, but you will also have a quick view on the project stats with an out-of-the-box dashboard presenting the number of Work items created and the number of Work items completed. Also, you can see the members in the end of the page.

The screenshot shows the Azure DevOps interface for the 'UtilityApp' project. The left sidebar includes links for Overview, Summary, Dashboards, Wiki, Boards, Repos, Pipelines, Test Plans, and Artifacts. The main content area features a welcome message 'Welcome to the project!' with a sub-instruction 'What service would you like to start with?' followed by buttons for Boards, Repos, Pipelines, Test Plans, and Artifacts. Below these buttons is the text 'or manage your services'. To the right, there's a 'Project stats' section with a note 'No stats are available at this moment. Setup a service to see project activity.' and a 'Members' section showing one member. The top navigation bar shows the path 'StudyJunction / UtilityApp / Overview / Summary' and includes a search bar, a 'Private' button, an 'Invite' button, and other standard UI elements.



Overview | Dashboards

In the dashboards page it's possible to create dashboards based on data from the work items, build history and commits done by the team. Each dashboard can be a Team Dashboard or a Project Dashboard, so you can target valuable data to each user and their roles.

The screenshot shows the Azure DevOps interface for the UtilityApp project. The left sidebar lists various project management sections: Overview, Summary, Dashboards (which is selected), Wiki, Boards, Repos, Pipelines, Test Plans, and Artifacts. The main content area displays the 'UtilityApp Team - Overview' page under the 'Dashboards' section. At the top, there are navigation links for StudyJunction, UtilityApp, Overview, and Dashboards, along with a search bar and several icons for refresh, edit, and more. A large central area features a cartoon illustration of two people holding up a large blue board with various charts and graphs. Below the illustration, the text reads: 'This dashboard doesn't have widgets just yet! Add one or more widgets to gain visibility into your team's progress.' A blue button labeled 'Add a widget' is visible at the bottom of this section.



Overview | Wiki

The wiki page present information in Markdown or simple text/HTML. You can also reference some code as a Wiki if you want to reuse some markdown files. This is very important to be created and maintained by your team to be more productive and distribute information easily.

Azure DevOps

StudyJunction / UtilityApp / Overview / Wiki

Search

UtilityApp

Overview

Summary

Dashboards

Wiki

Boards

Repos

Pipelines

Test Plans

Artifacts

Project settings

Get everyone on the same page

Create wiki pages to enable your team members collaborate

Create project wiki

Publish code as wiki

Learn more

An illustration of a person painting on a large easel while a dog sits nearby.



Boards | Work items

The Work items page gives a quick overview about work items not closed in your project. You can easily create a new work item, define new queries, specify columns, delete them and recover from the recycle bin or import new work items.s

The screenshot shows the Azure DevOps interface for the 'UtilityApp' project. The left sidebar includes links for Overview, Boards, Work items (which is the active tab), Boards, Backlogs, Sprints, Queries, Delivery Plans, Repos, Pipelines, Test Plans, Artifacts, and Project settings. The main area is titled 'Work items' and features a 'Recently updated' section with a large orange button containing a white 'L' and a circular arrow icon. Below this are filter options for 'Filter by keyword', 'Types', 'Assigned to', 'States', 'Area', and 'Tags'. A search bar and various navigation icons are at the top right.

StudyJunction / UtilityApp / Boards / Work items

Search

UtilityApp

Overview

Boards

Work items

Recently updated

New Work Item

Open in Queries

Column Options

Import Work Items

Filter by keyword

Types

Assigned to

States

Area

Tags

L

Find recently updated items

View items that have been recently updated.

Learn more about work items



Boards | Boards

The Boards page shows the stories in different boards and in different levels. By default, you will see a board in User Story or Feature level, but you can configure to show at Epic level or configure other types of Work Items.

The screenshot shows the Azure DevOps Boards page for the UtilityApp Team. The left sidebar menu includes options like Overview, Boards (selected), Work items, Backlogs, Sprints, Queries, Delivery Plans, Repos, Pipelines, Test Plans, Artifacts, and Project settings. The main area displays a Kanban board with columns for New, Active, Resolved, and Closed work items. The New column contains a 'New item' button and a search icon. The Active, Resolved, and Closed columns are currently empty. The top navigation bar shows the project path: StudyJunction / UtilityApp / Boards / Boards. The top right corner features a search bar, filter icons, and a 'BP' button.



Repos | Files

The Files page offers you a view of the files available in your project, organized by project/branches. There you can select the branch you want to look for, navigate to other projects and here you even can access the Clone button. The clone button allows you to clone with different protocols, so you can start working from your machine.

The screenshot shows the Azure DevOps interface for the 'UtilityApp' repository. The left sidebar includes links for Overview, Boards, Repos (selected), Files, Commits, Pushes, Branches, Tags, Pull requests, Pipelines, and Project settings. The main content area displays the message 'UtilityApp is empty. Add some code!' and provides options for cloning the repository via HTTPS or SSH, generating Git credentials, and pushing an existing repository from the command line. It also includes sections for importing a repository and initializing the main branch with a README or gitignore file.

Azure DevOps

StudyJunction / UtilityApp / Repos / Files / UtilityApp

Search

UtilityApp

UtilityApp

Overview

Boards

Repos

Files

Commits

Pushes

Branches

Tags

Pull requests

Pipelines

Project settings

UtilityApp is empty. Add some code!

Clone to your computer

HTTPS SSH https://StudyJunction@dev.azure.com/StudyJunction/UtilityApp/_git

Generate Git Credentials

Having problems authenticating in Git? Be sure to get the latest version [Git for Windows](#) or our plugins for [IntelliJ](#), [Eclipse](#), [Android Studio](#) or [Windows command line](#).

Clone in VS Code

Push an existing repository from command line

HTTPS SSH

git remote add origin https://StudyJunction@dev.azure.com/StudyJunction/UtilityApp/_git/UtilityApp

Import a repository

Import

Initialize main branch with a README or gitignore

Add a README Add a .gitignore: None Initialize



Repos | Commits

The Commits page lists all the commits made to a repository. You can use the search bar to filter commits or the sidebar to recover one specific version. The sidebar also shows the commit timeline.

The screenshot shows a software interface for managing repositories and commits. The top navigation bar includes 'StudyJunction / UtilityApp / Repos / Commits / UtilityApp'. A search bar is at the top right, along with various icons for filtering and searching. On the left, a sidebar titled 'dev' contains icons for user management, adding new items, charts, code review, pull requests, and more. The main area is titled 'Commits' and displays a timeline of commits. A vertical timeline on the left shows commit history with blue dots. The commit list includes:

Commit	Pull Request	Status
Merged PR 3: templates-build.yml added 9eb227d5 Bhaumik Patel Today at 9:30 AM	3	succeeded
code correct da753aec Bhaumik Patel Today at 9:26 AM	3	
format 6f36df6b Bhaumik Patel Today at 9:24 AM	3	succeeded
variables set f8d27e2c Bhaumik Patel Today at 9:22 AM	3	
Finaly pipeline 4dddf0d08 Bhaumik Patel Today at 8:59 AM	3	
parameters 17730d25 Bhaumik Patel Today at 8:47 AM	3	succeeded
displayName 318e0b99 Bhaumik Patel Today at 8:45 AM	3	
parameters pass	3	



Pipelines | Pipelines

It is a workflow that defines how our test, build, and deployment steps are run.

The screenshot shows the Azure Pipelines interface for the 'StudyJunction / UtilityApp / Pipelines' project. The left sidebar includes icons for repository, pipeline, release, test, artifact, and more. The main area displays a list of recently run pipelines:

Pipeline	Last run	Duration
Web App	#20220727.1 • Merged PR 3: templates-build.yml added Individual CI for dev	3h ago 21s
Password Generator API Pipeline	#20220727.2 • Merged PR 3: templates-build.yml added Individual CI for dev	3h ago 33s
Weather Forecast API Pipeline	#20220727.1 • Merged PR 3: templates-build.yml added Individual CI for dev	3h ago 35s
UtilityApp	#20220727.3 • parameters Individual CI for master	4h ago 21s



Pipelines | Environment

It is a collection of resources, where you deploy your application. It contains one or more virtual machines, containers, web apps, etc.

StudyJunction / UtilityApp / Pipelines / Environments

Search

BP

Create your first environment

Manage deployments, view resource status and get full end to end traceability

Create environment



Pipelines | Releases

Release pipelines in Azure
Pipelines help your team continuously deliver software to your customers at a faster pace and with lower risk. You can fully automate the testing and delivery of your software in multiple stages all the way to production.

The screenshot shows the Azure Pipelines interface for the 'StudyJunction / UtilityApp / Pipelines / Releases' path. A search bar and various navigation icons are at the top. On the left is a sidebar with icons for Update, New Pipeline, Pipelines, Artifacts, Pipelines, Test, and Pipelines. The main area displays a cartoon illustration of a person and a dog launching a rocket from a launch pad. Below the illustration, the text 'No release pipelines found' is centered, followed by the subtext 'Automate your release process in a few easy steps with a new pipeline' and a prominent blue 'New pipeline' button.



Pipelines | Library

Variable groups store values across multiple pipelines. You

The screenshot shows the Azure Pipelines Library interface. The top navigation bar includes the project name "StudyJunction / UtilityApp / Pipelines / Library". On the right, there's a search bar, a ribbon menu, and a "BP" button. The main area is titled "Library" and has tabs for "Variable groups" (which is selected), "Secure files", and "Variable group" (with a plus sign icon). Below these are icons for "Variable group", "Secure file", "Variable group", "Variable group", "Variable group", "Variable group", "Variable group", and "Variable group". A "Security" and "Help" link are also present. To the right, a large button labeled "+ Variable group" with a "(x)" icon is highlighted with a red box. Below it, text reads "New variable group" and "Create groups of variables that you can share across multiple pipelines." A link "Learn more about variable groups." is at the bottom.

StudyJunction / UtilityApp / Pipelines / Library

Search

Library

Variable groups Secure files + Variable group Security Help

+ Variable group

New variable group

Create groups of variables that you can share across multiple pipelines.

Learn more about variable groups.



Pipelines | Task groups

A task group allows you to encapsulate a sequence of tasks, already defined in a build or a release pipeline, into a single reusable task that can be added to a build or release pipeline, just like any other task.

The screenshot shows the 'Task groups' page in the Azure Pipelines interface. The top navigation bar includes 'StudyJunction / UtilityApp / Pipelines / Task groups'. On the left, there's a vertical toolbar with icons for 'Task groups' (highlighted), 'Import', 'Security', and other pipeline management options. The main area displays a large 'Import a task group' button with a 'Task group' icon above it. Below the button, the text reads: 'Standardize and centrally manage common build and deployment steps for your applications.' A prominent blue 'Import a task group' button is centered at the bottom of the section. To its right, a link says 'Learn more about task groups'.



Pipelines | Deployment groups

A deployment group is a logical set of deployment target machines that have agents installed on each one. Deployment groups represent the physical environments; for example, "Dev", "Test", or "Production" environment. In effect, a deployment group is just another grouping of agents, much like an agent pool.

The screenshot shows the 'Deployment groups' page in the Azure Pipelines interface. The navigation bar at the top includes 'StudyJunction / UtilityApp / Pipelines / Deployment groups'. On the right, there are icons for search, refresh, help, and a 'BP' button. The main area has a title 'Deployment groups' and a table with columns 'Name', 'Target status', and 'Deployment status'. A single row is shown for 'LocalPC', which has '1 Online' target status and '1 Never deployed' deployment status. The left sidebar contains icons for 'Groups' (selected), 'Available pools', 'New', 'Security', 'Help', and other navigation options.

Name	Target status	Deployment status
LocalPC	1 Online	1 Never deployed