TEKsystems Global Services | 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

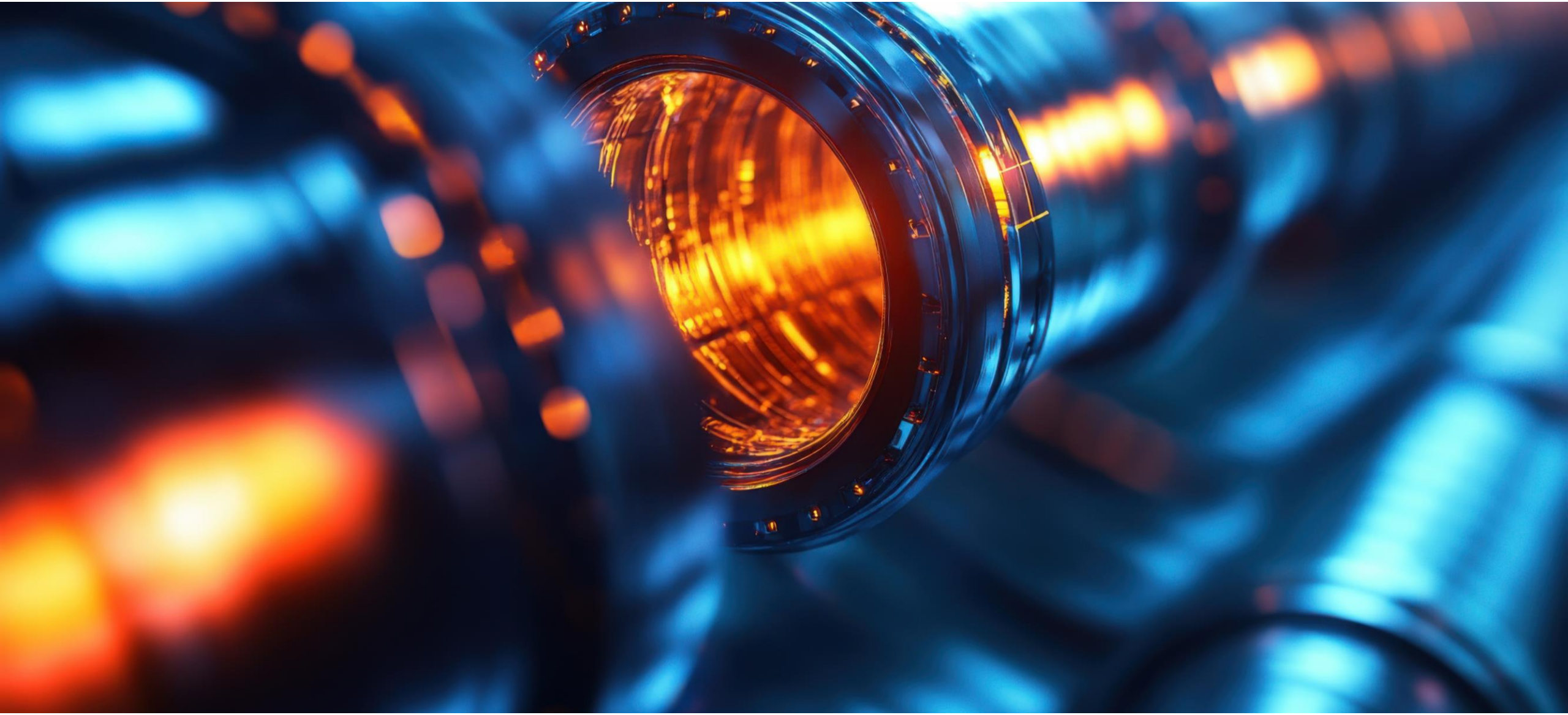
3

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

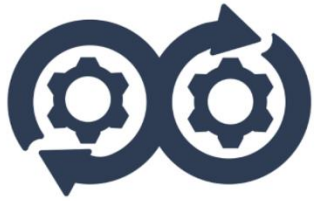
Expertise

- Cloud
- Automation
- CICD
- Docker
- Kubernetes

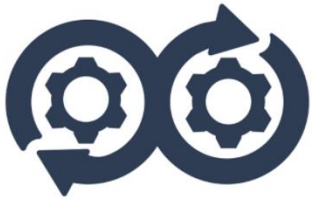
Introductions

- 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



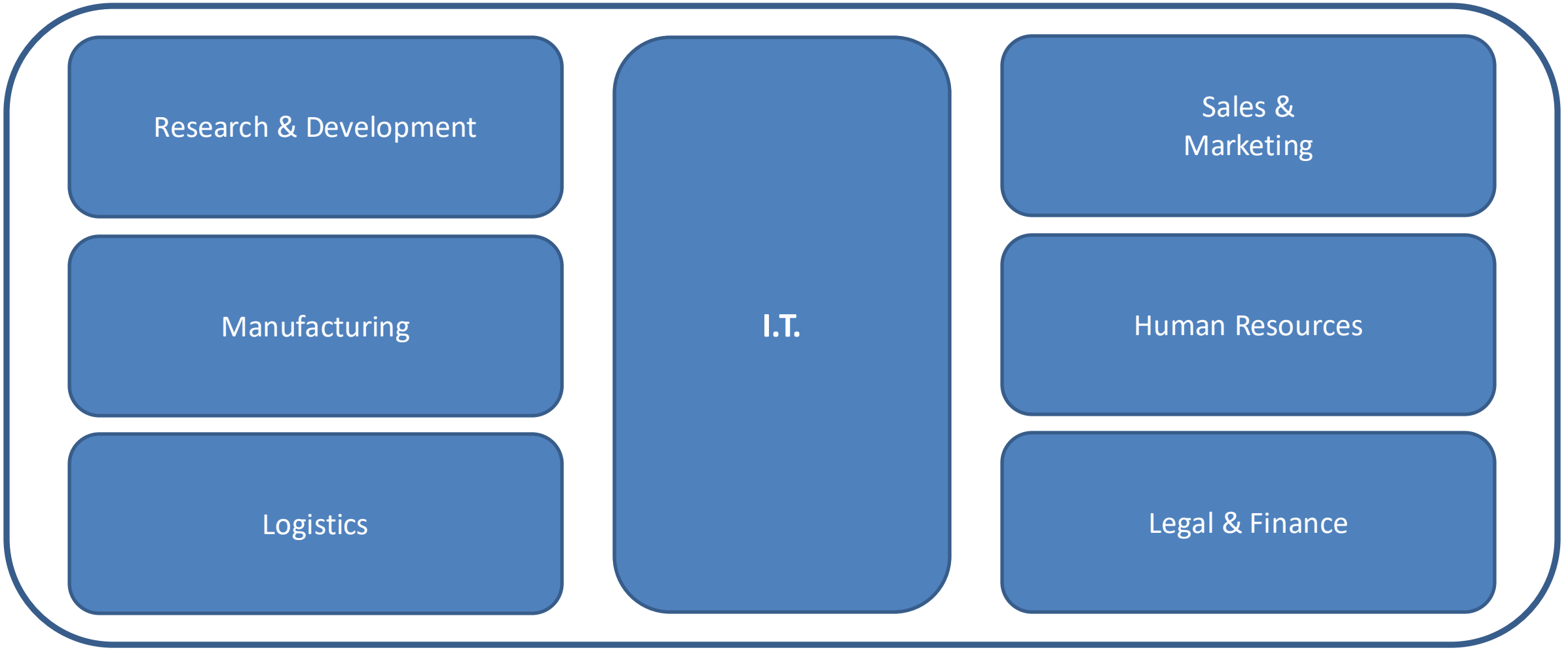
Traditional Org Structure



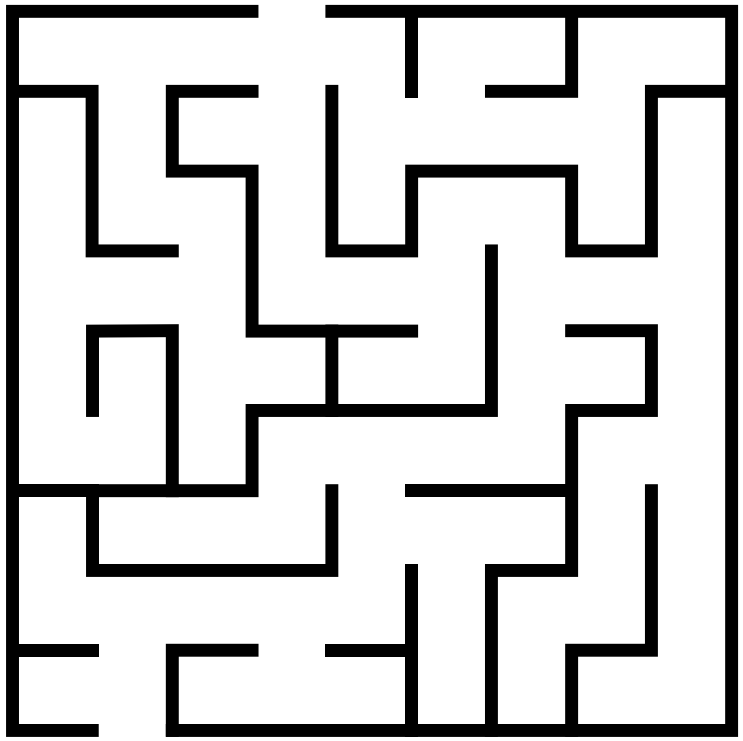
POP QUIZ: Organization

How is your company organized?

Arranged by Functional Area

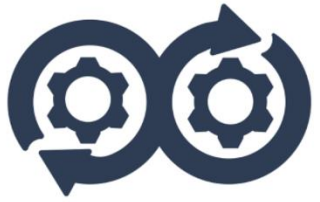


Complex Deployment Pipeline



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

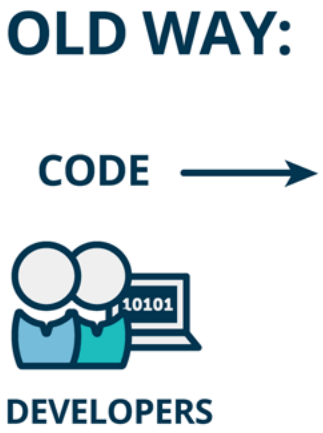
Software Development Life Cycle



POP QUIZ: Software Development Life Cycle

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

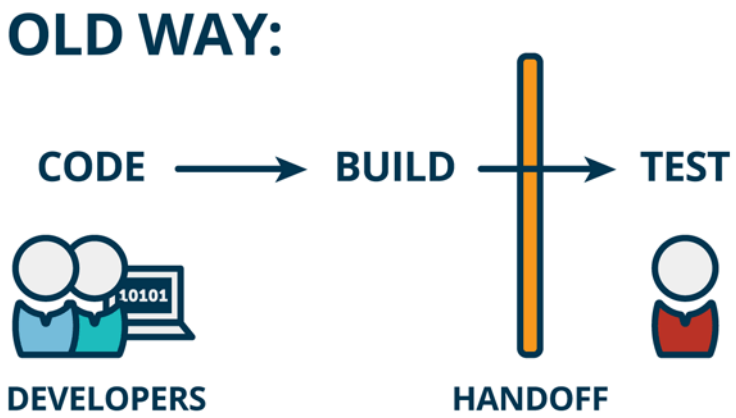
Dev Ops & QA



Developers build application

- Runtime
- Libraries
- Dependencies

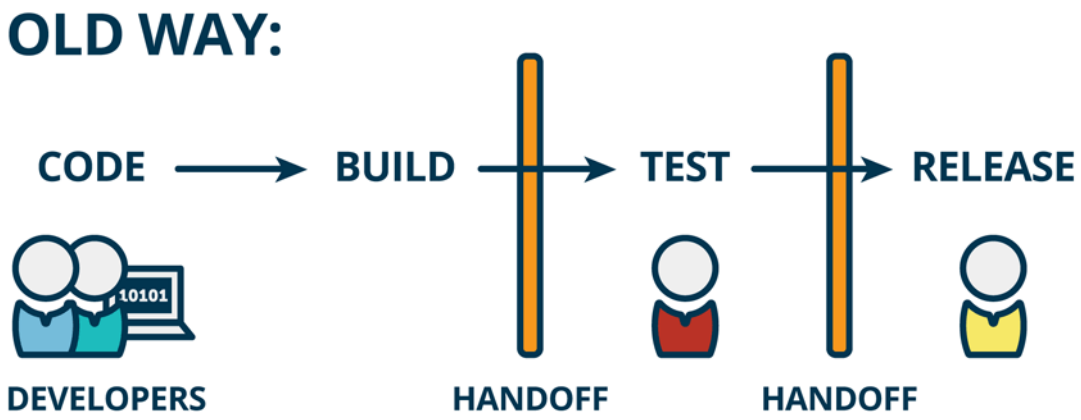
Dev Ops & QA



Handed off to Test

- Unit tests
- Integration tests
- Was story completed?

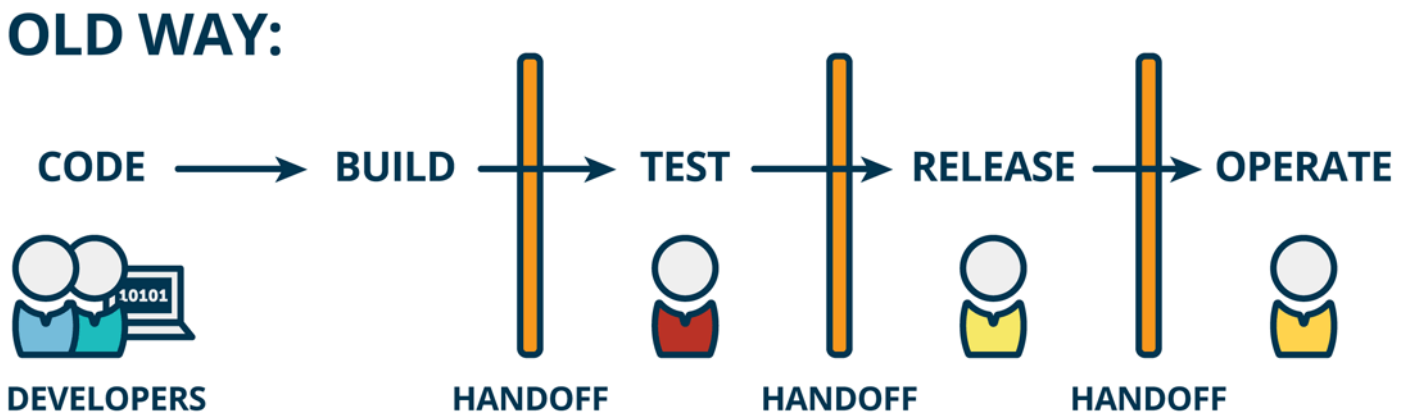
Dev Ops & QA



Deployed to Production

- Deployment tooling
- Artifacts
- Services

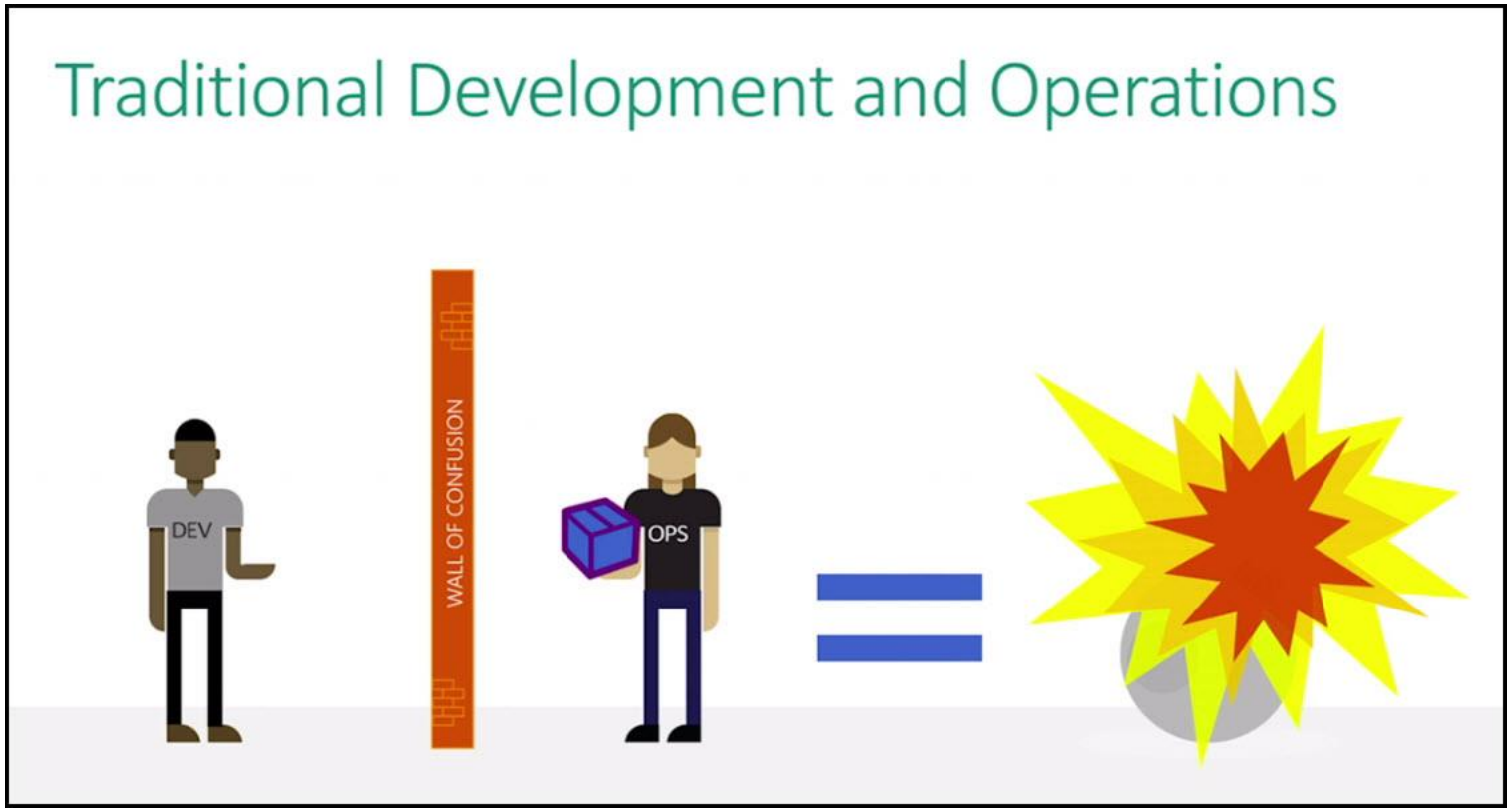
Dev Ops & QA



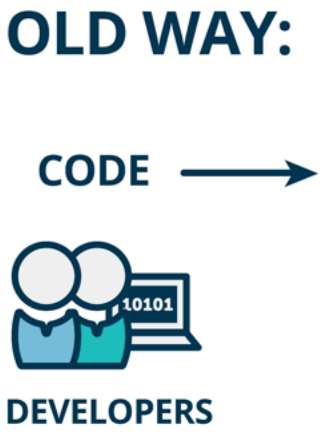
Ops now manages applications

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

Dev vs Ops



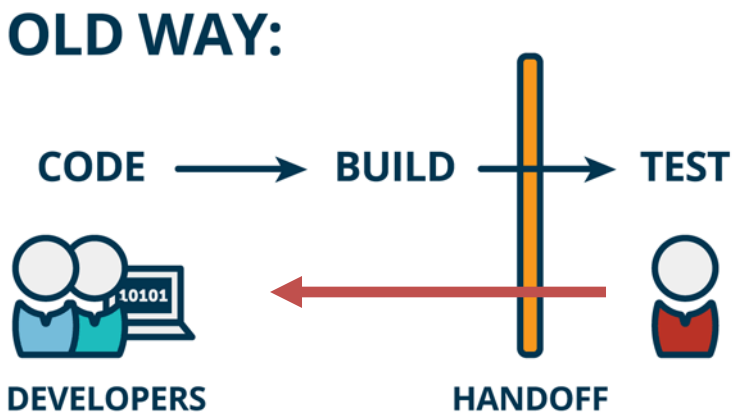
What really happens



Developers build application

- Runtime
- Libraries
- Dependencies

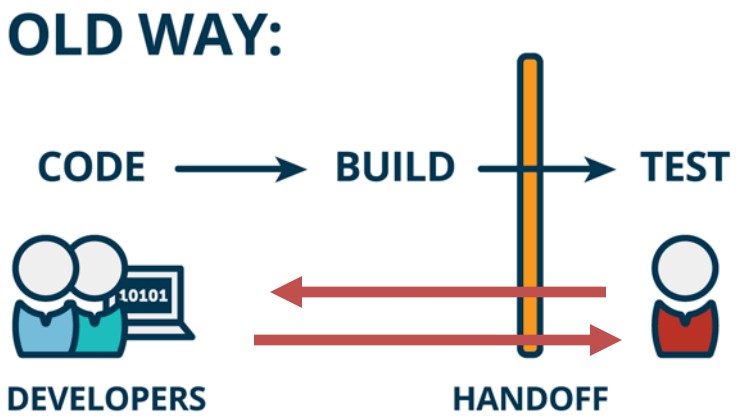
What really happens



Handed off to Test

- Unit tests
- Integration tests
- Was story completed?

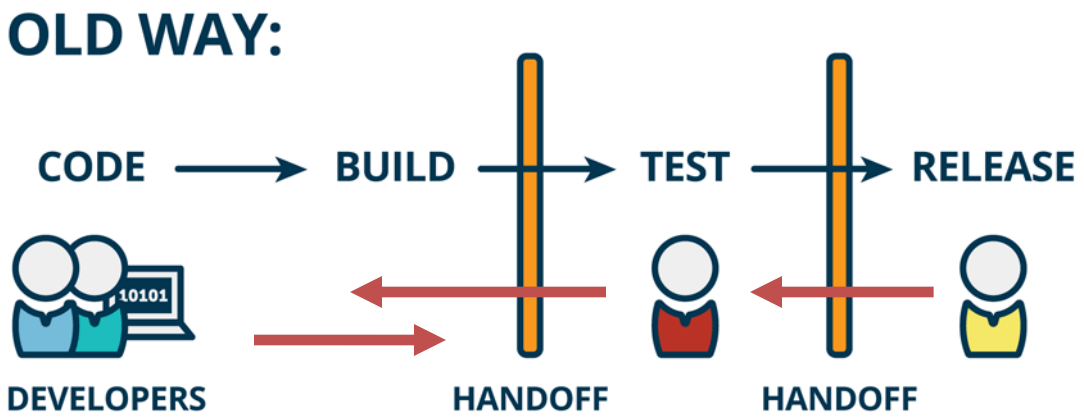
What really happens



Handed off to Test

- Unit tests
- Integration tests
- Was story completed?

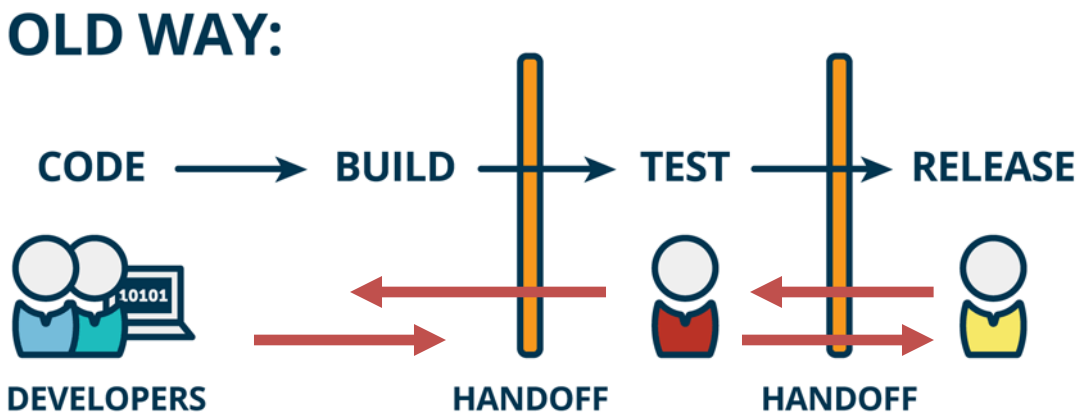
What really happens



Deployed to Production

- Deployment tooling
- Artifacts
- Services

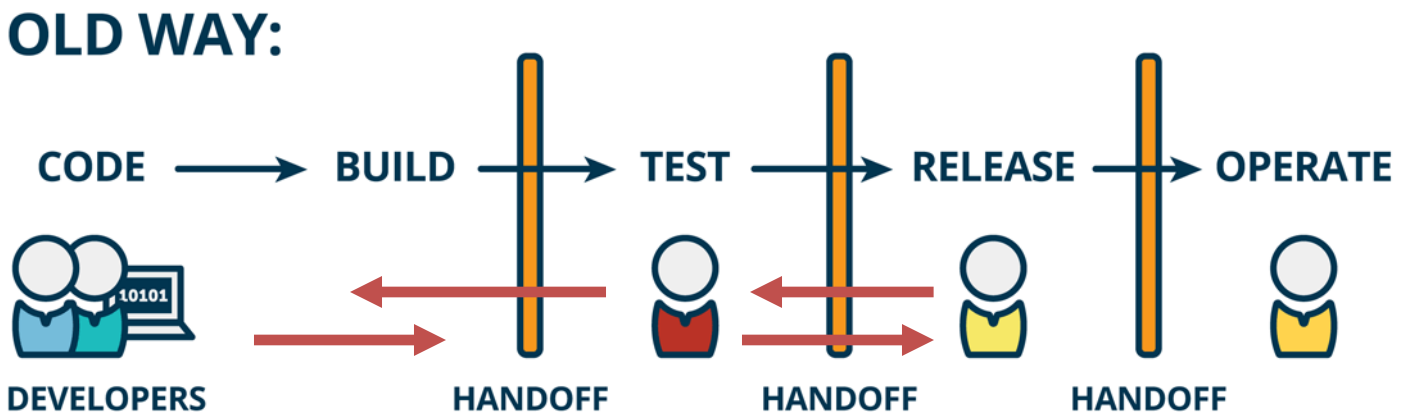
What really happens



Deployed to Production

- Deployment tooling
- Artifacts
- Services

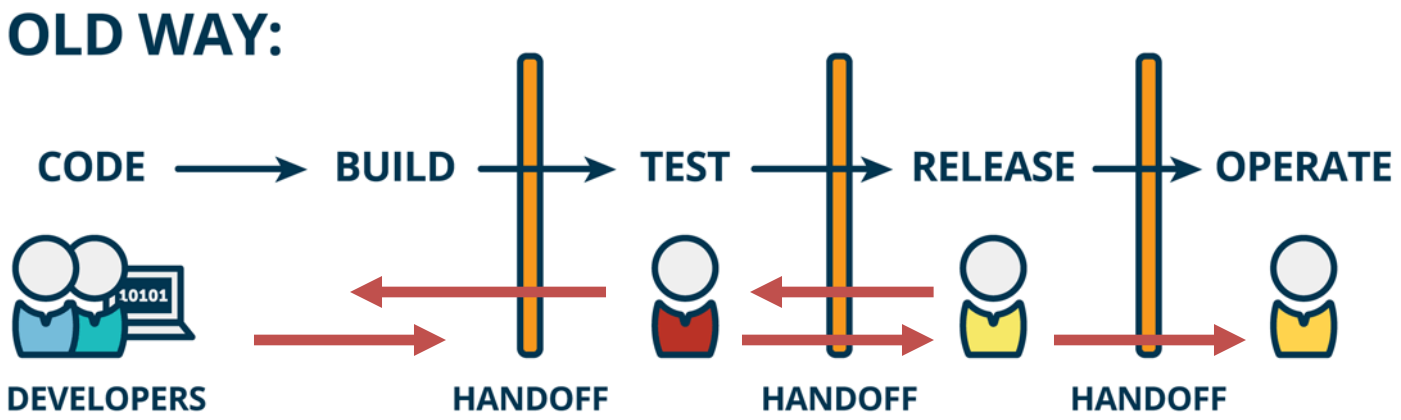
What really happens



Ops now manages applications

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

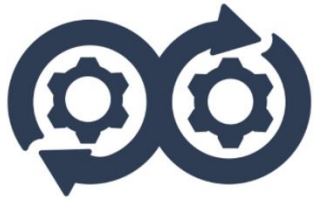
What really happens



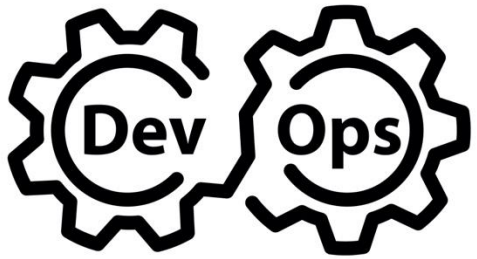
Ops now manages applications

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

What is DevOps?



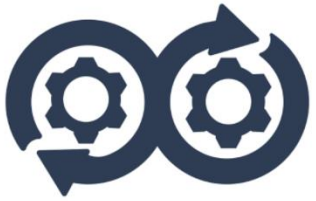
C.A.L.M.S



- 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!

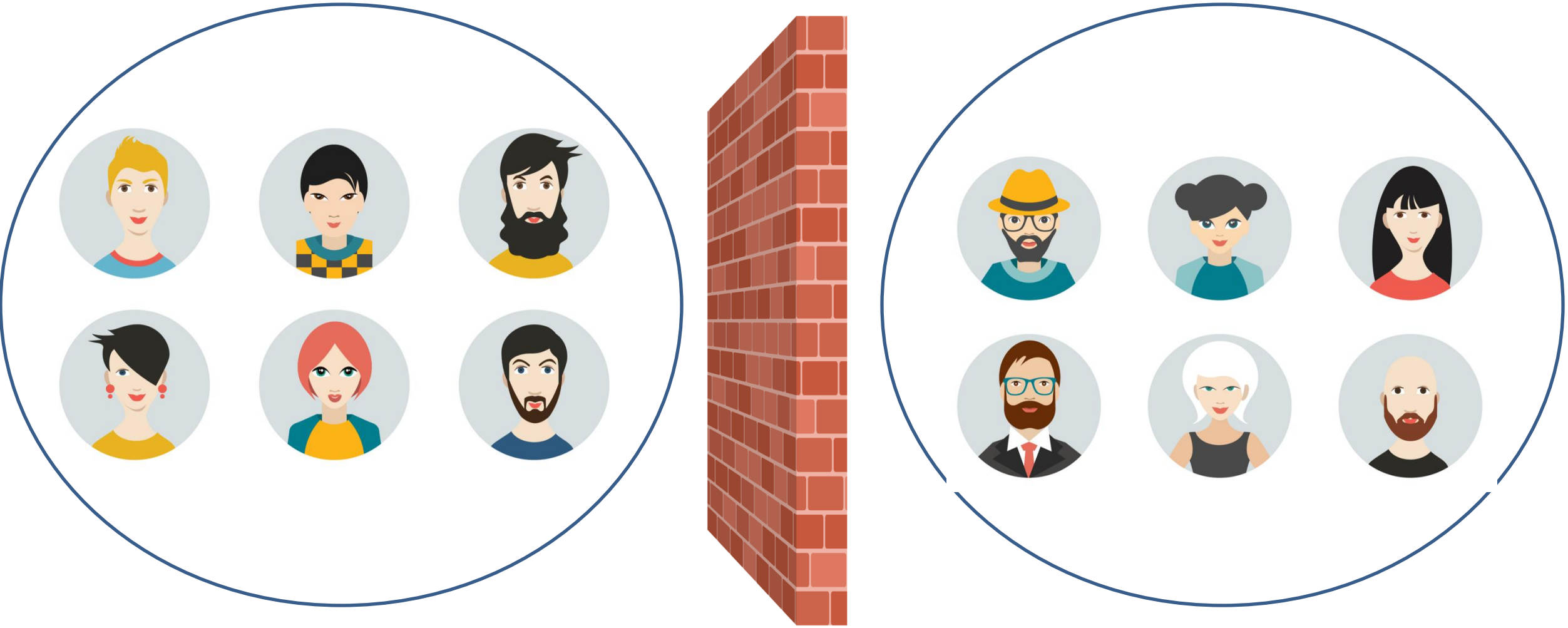
POP QUIZ: Operations responsibilities?

As an operations engineer what are your primary responsibilities?

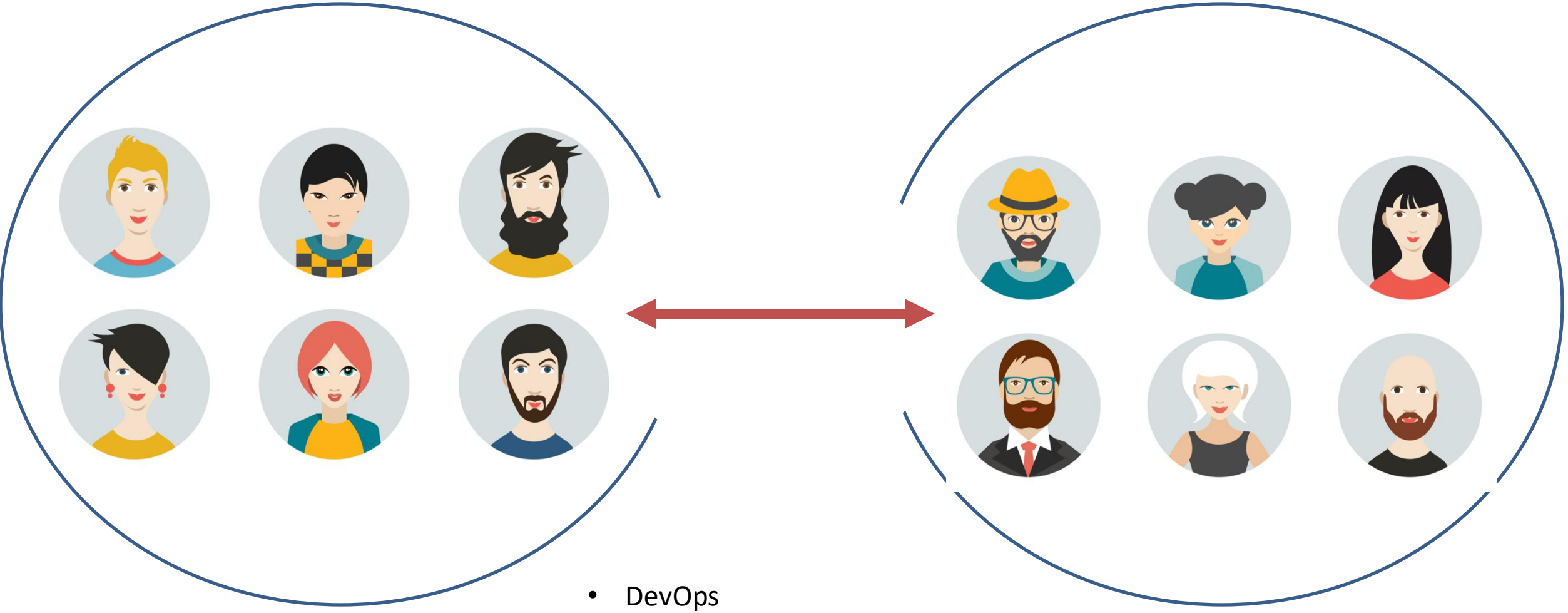
People - Operations

- Ops
  - Stability
  - Deliver services
  - Resist change!

Silos



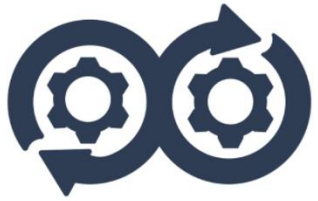
Collaboration!



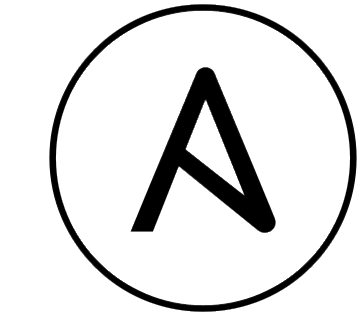
- DevOps
  - Two teams
  - Processes for communication/collaboration

Automation:

Repeatability by automating activities.



Automation tools



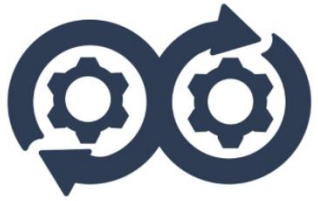
ANSIBLE

CHEF

puppet

HashiCorp Terraform

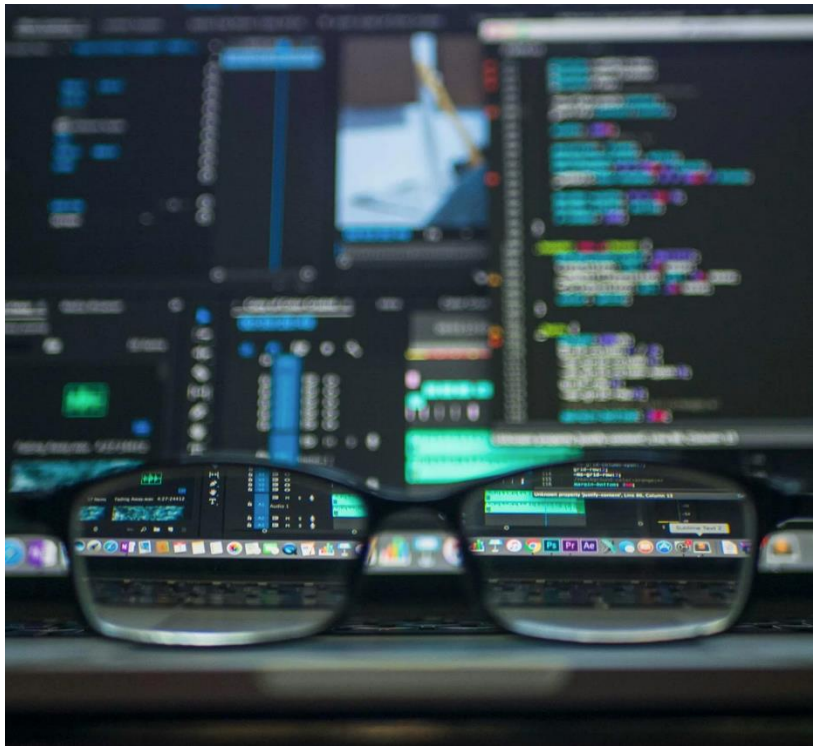
DevSecOps and Policy-as-Code



Introduction to DevSecOps

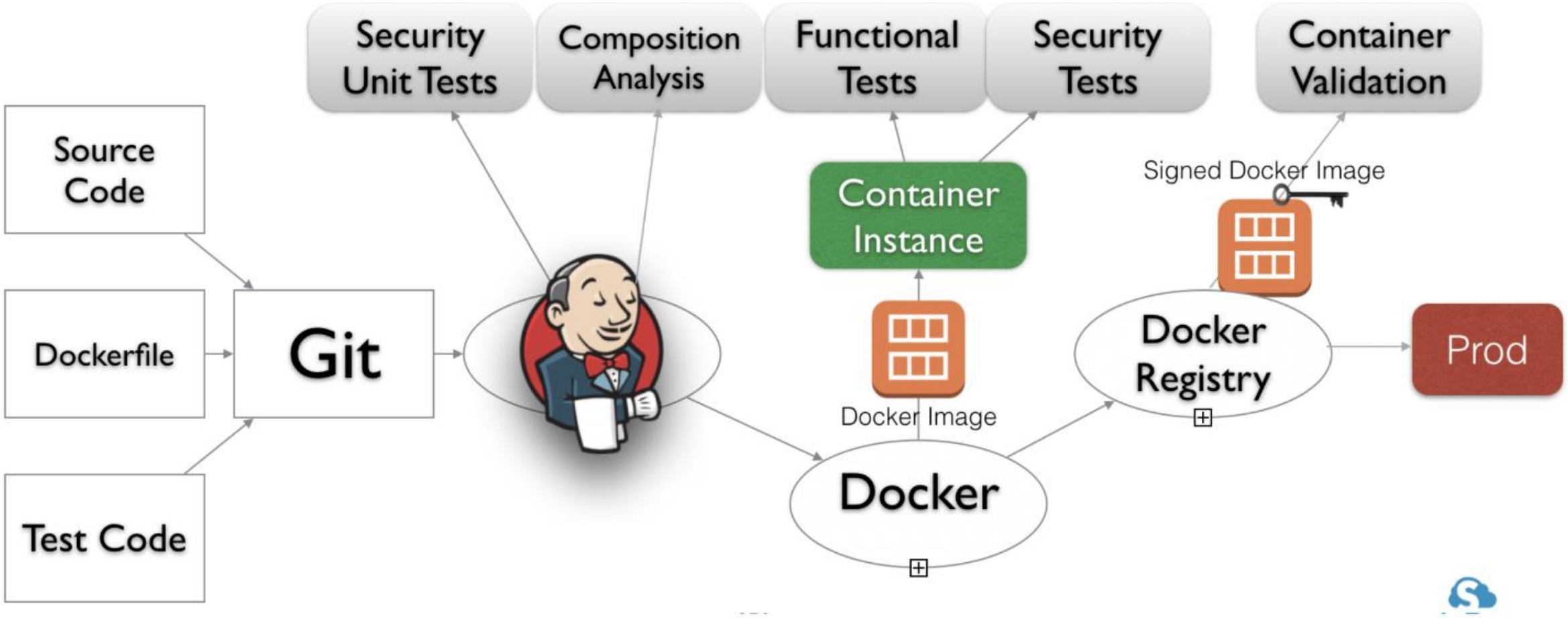
Continuously assessing security measures within the development cycle to gauge their effectiveness and foster ongoing improvement.

What is DevSecOps?

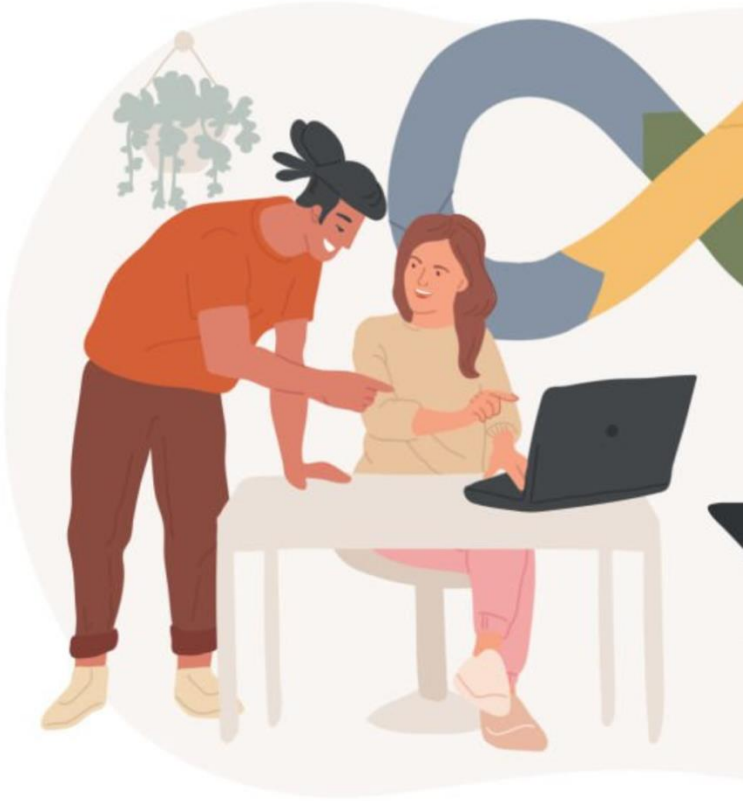


- **DevSecOps** integrates security practices into DevOps, making security a shared responsibility throughout the software development lifecycle.
- **Security** is embedded at every stage, from code development to deployment.
- **Goal:** Catch vulnerabilities early and ensure continuous protection.

Example DevSecOps Architecture



Key principles of DevSecOps



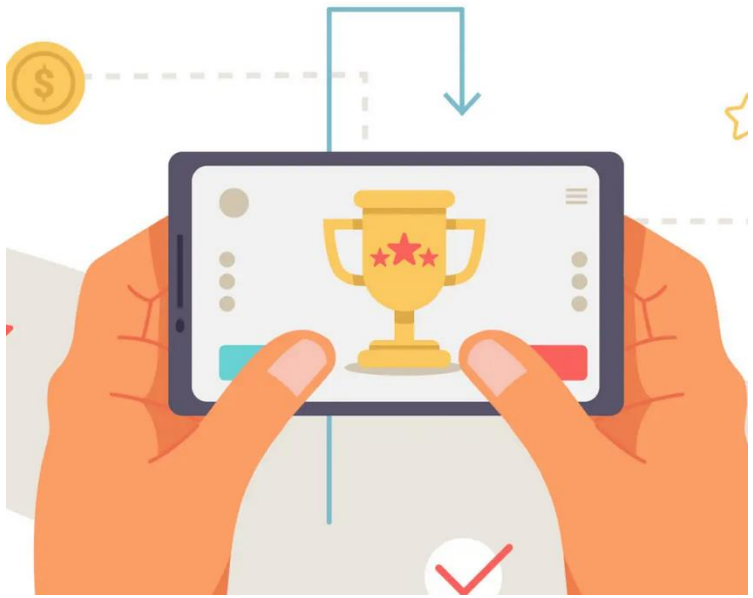
- **Continuous security integration**, automating security checks for faster, safer releases.
- **Close collaboration** among development, security, and operations teams for proactive threat mitigation.
- **Real-time monitoring and feedback loops**, allowing teams to assess and improve security measures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

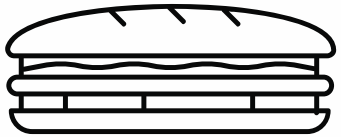
Infrastructure as Code

Provisioning infrastructure through software to achieve consistent and predictable environments.

Core Concepts

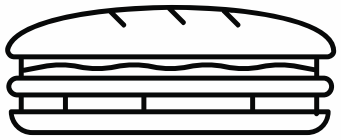
- Defined in code
- Stored in source control
- Declarative or imperative

Imperative



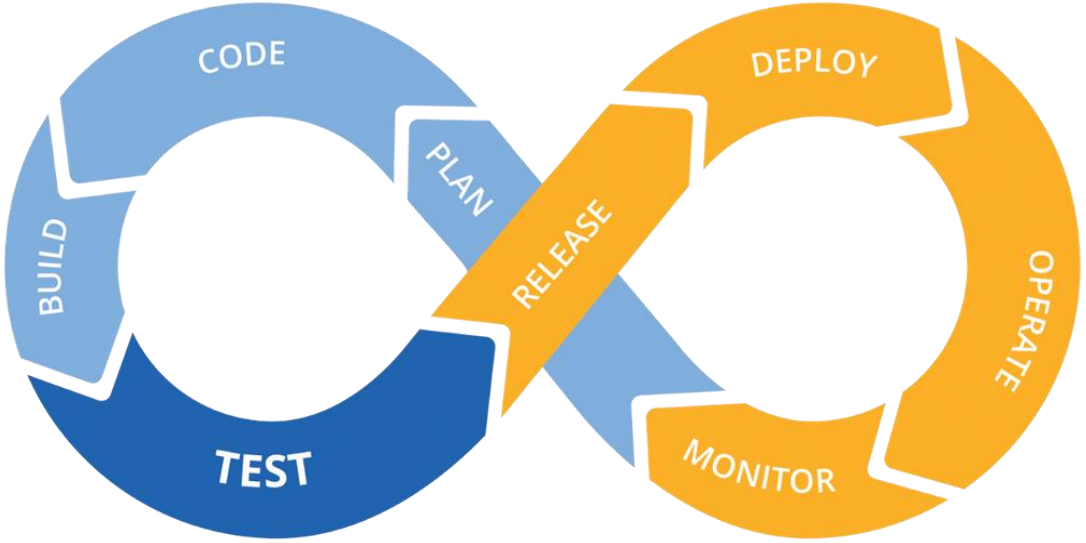
```
# Make a sandwich
get bread
get mayo
get turkey
get lettuce

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

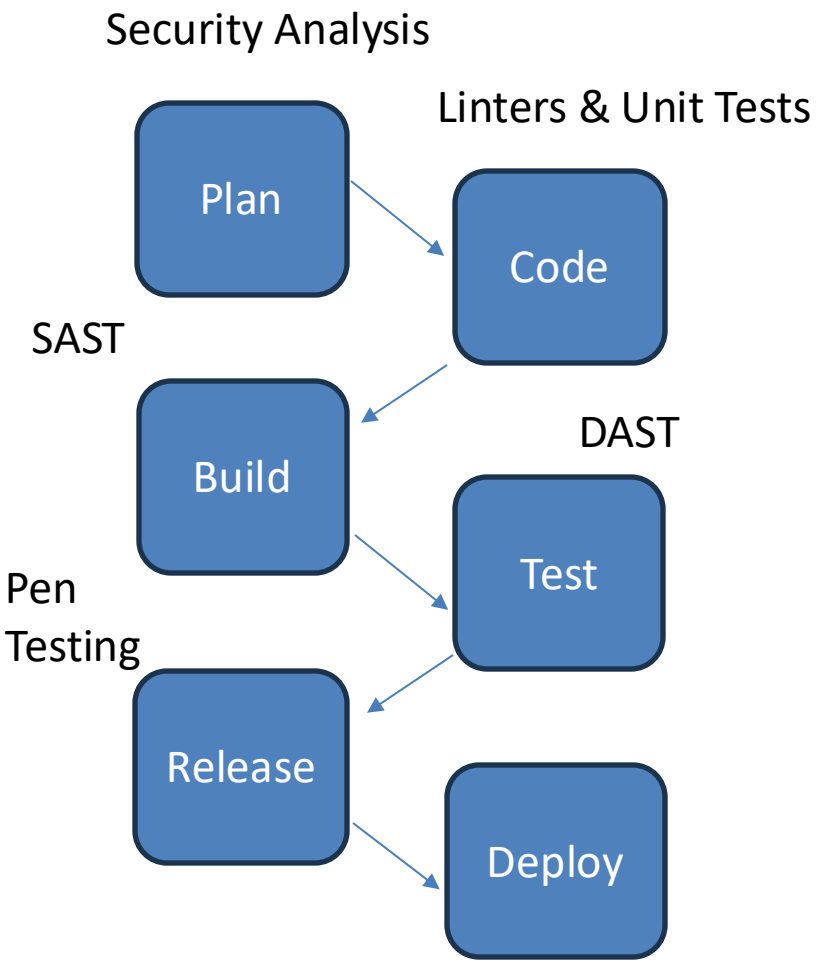
Early Detection: Vulnerabilities are identified before reaching production.

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



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

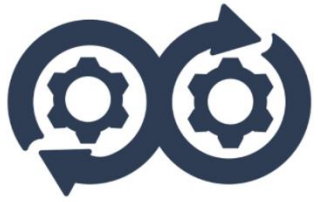
Dependency Scanning: Checks third-party libraries for known vulnerabilities to ensure secure dependencies.

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

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

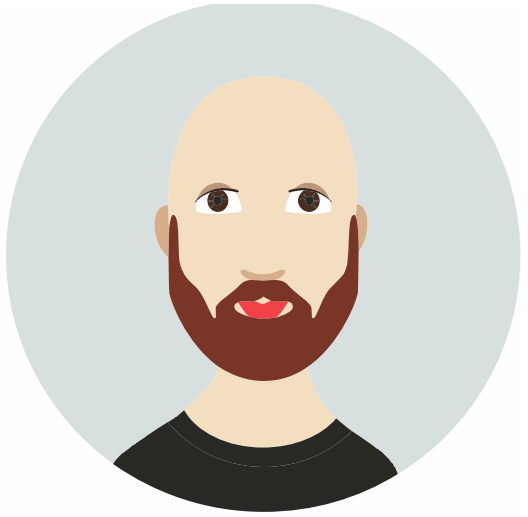
Dynamic Application Security Testing (DAST): Tests the running application for security weaknesses from an attacker's perspective.

Case Study: Media Corporation



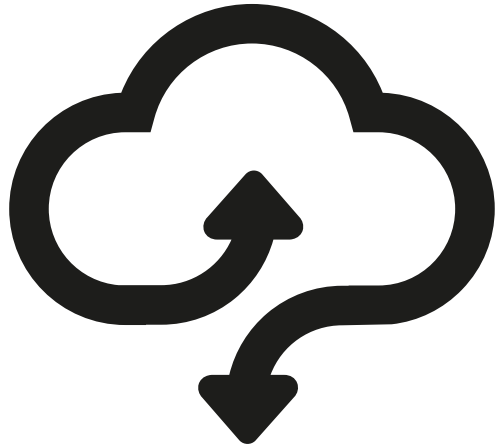
Git

Git



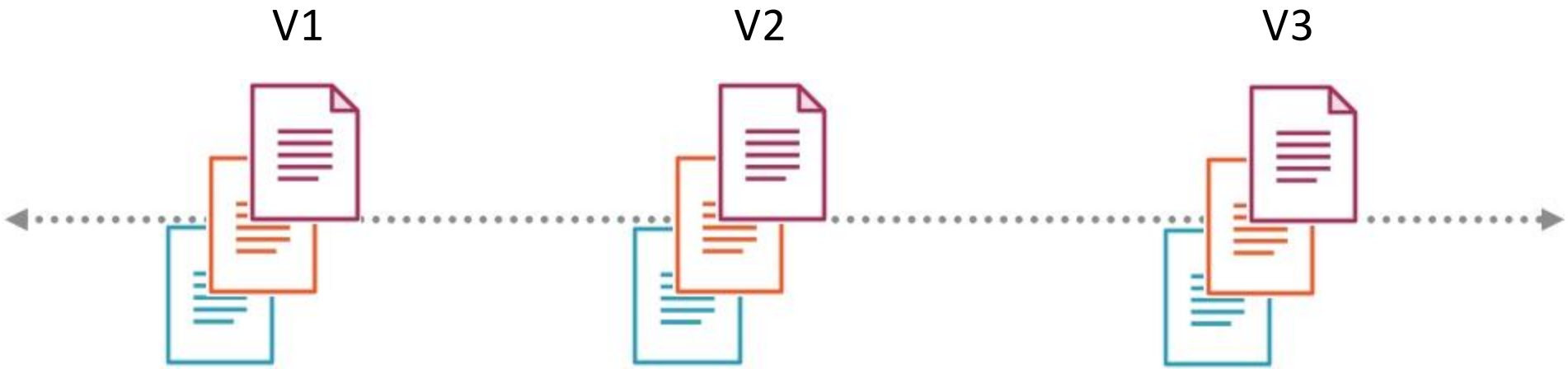
git

Git



What is Git?

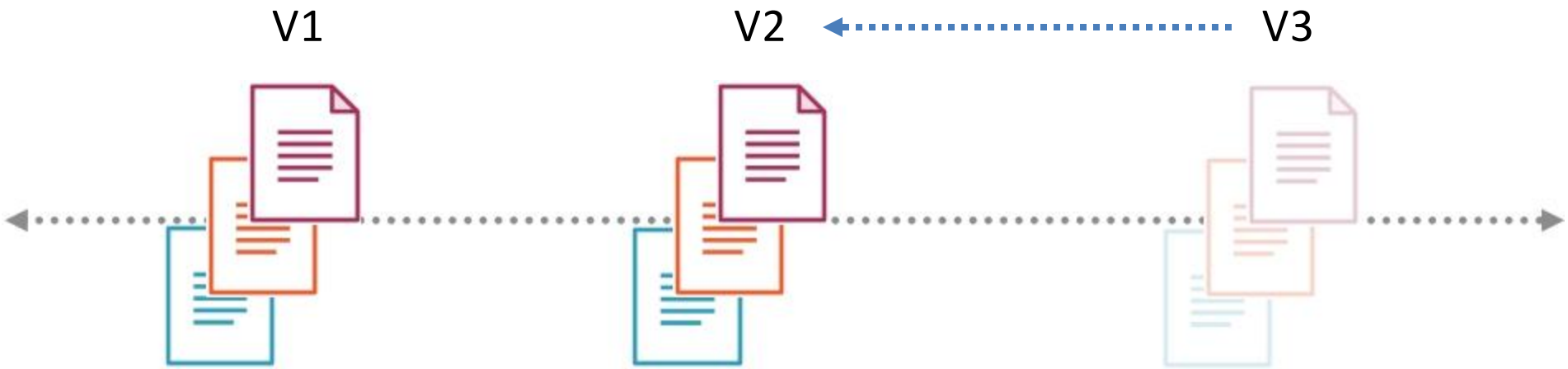
Version Control System



What is Git?

Version Control System

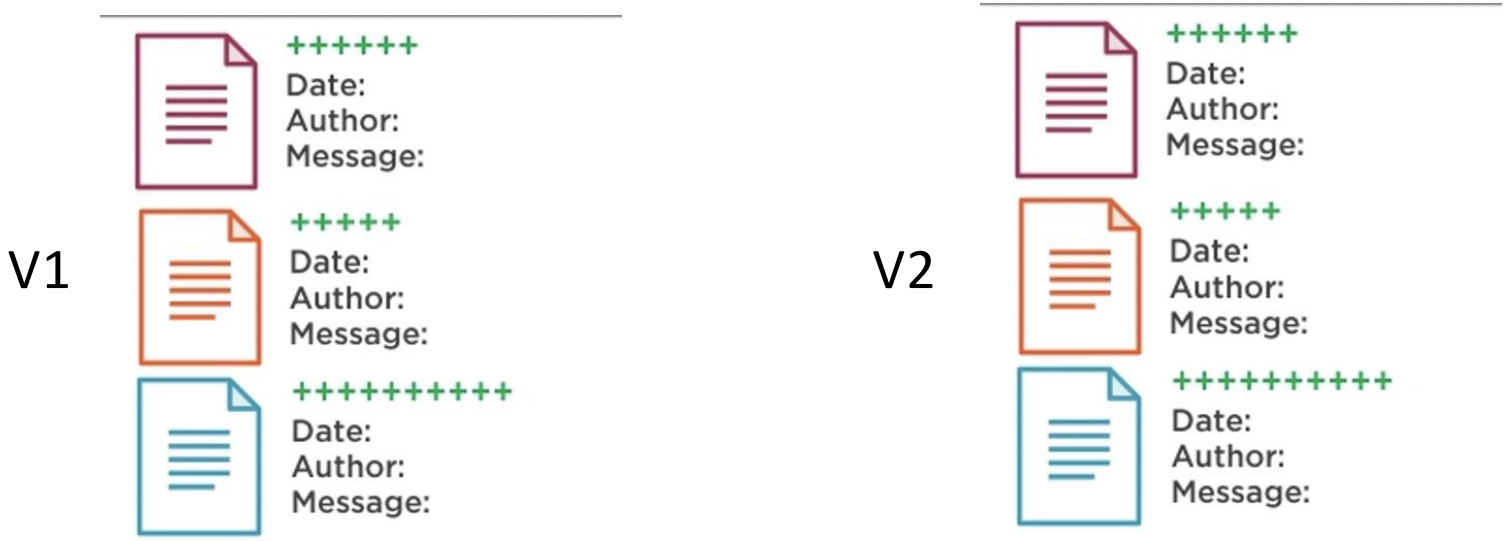
- Software designed to record changes to files made over time
- Ability to revert back to a previous file version or project version



What is Git?

Version Control System

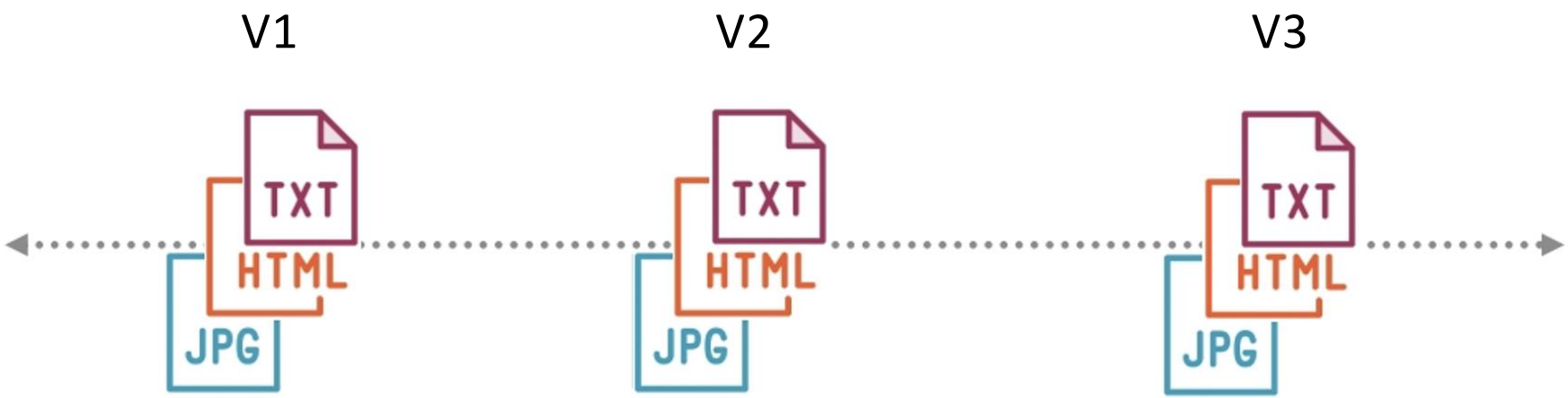
- Software designed to record changes to files made over time
- Ability to revert back to a previous file version or project version
- Compare changes made to files from one version to another



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

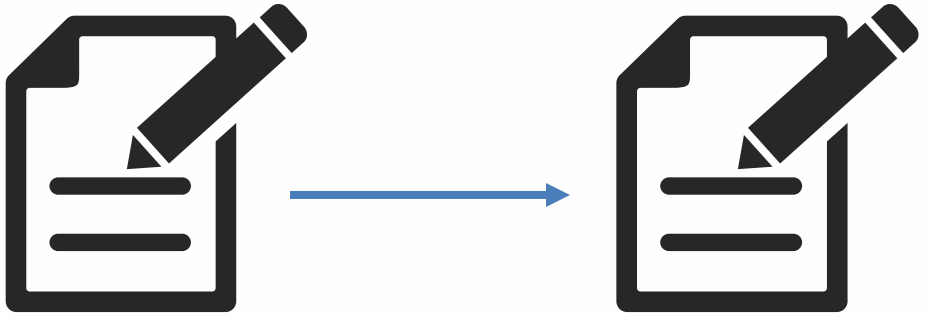
WIRED BRAIN
COFFEE

Git use-case

Latte

Git use-case

Latte



POP QUIZ: Dangers of copying files

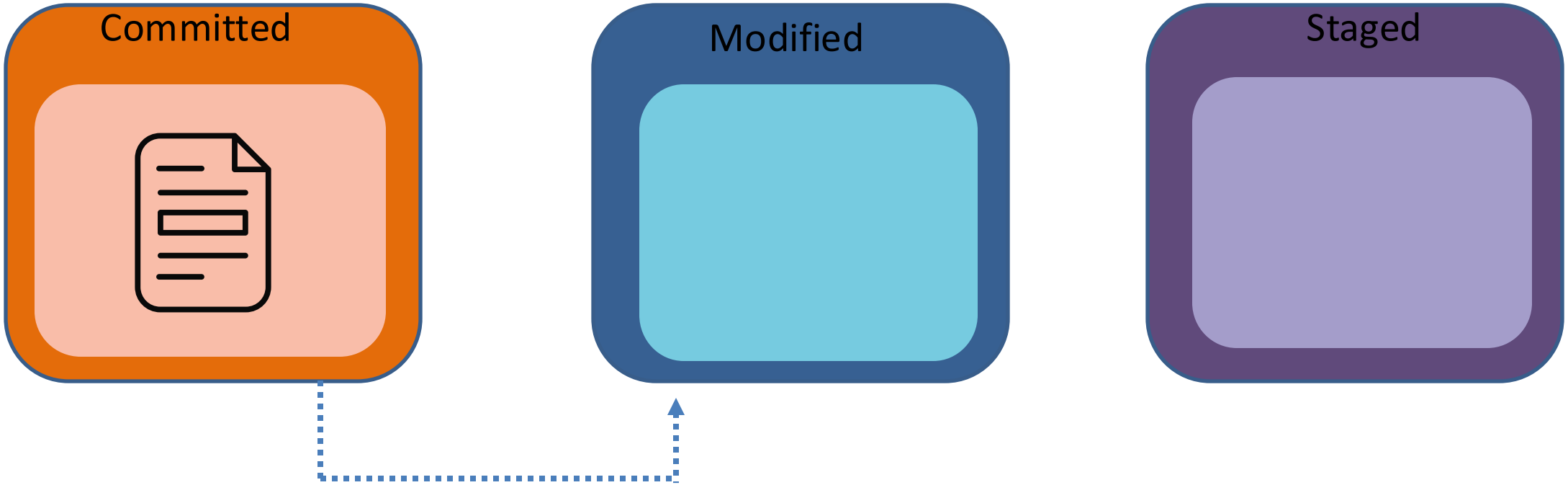
What are some dangers of Steve's current process?

POP QUIZ: Dangers of copying files

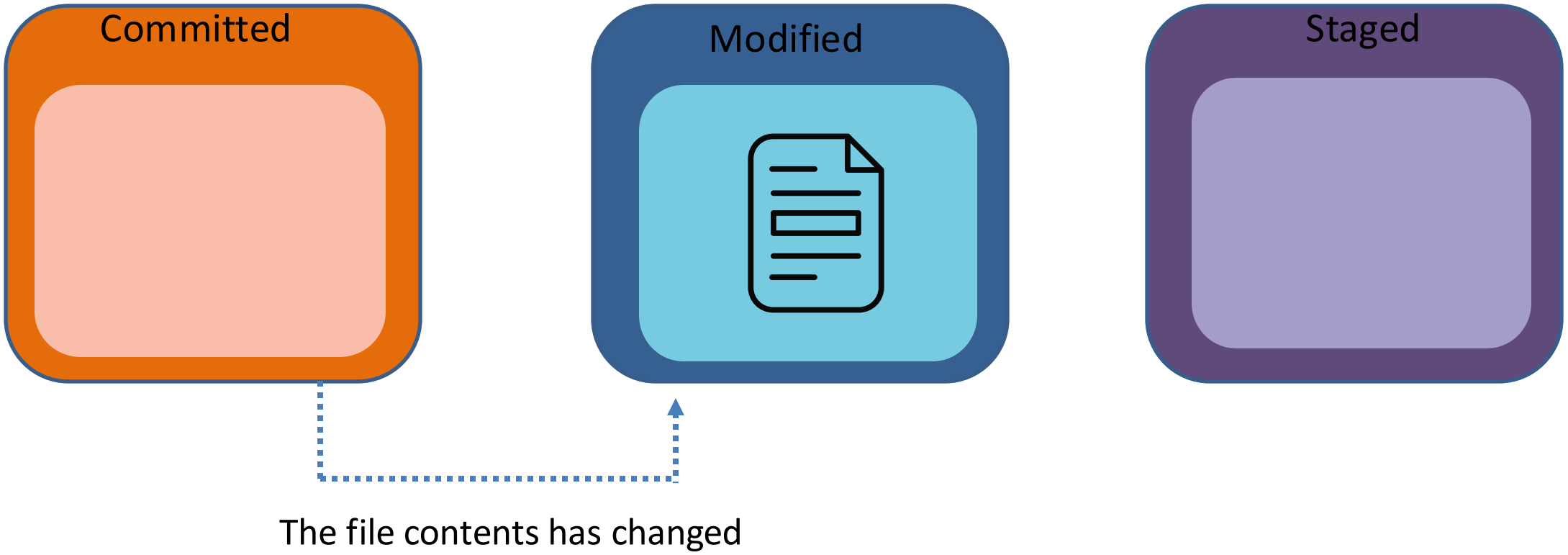
What are some dangers of Steve's current process?

- Files live in one location
- Accidentally overwrite existing recipes

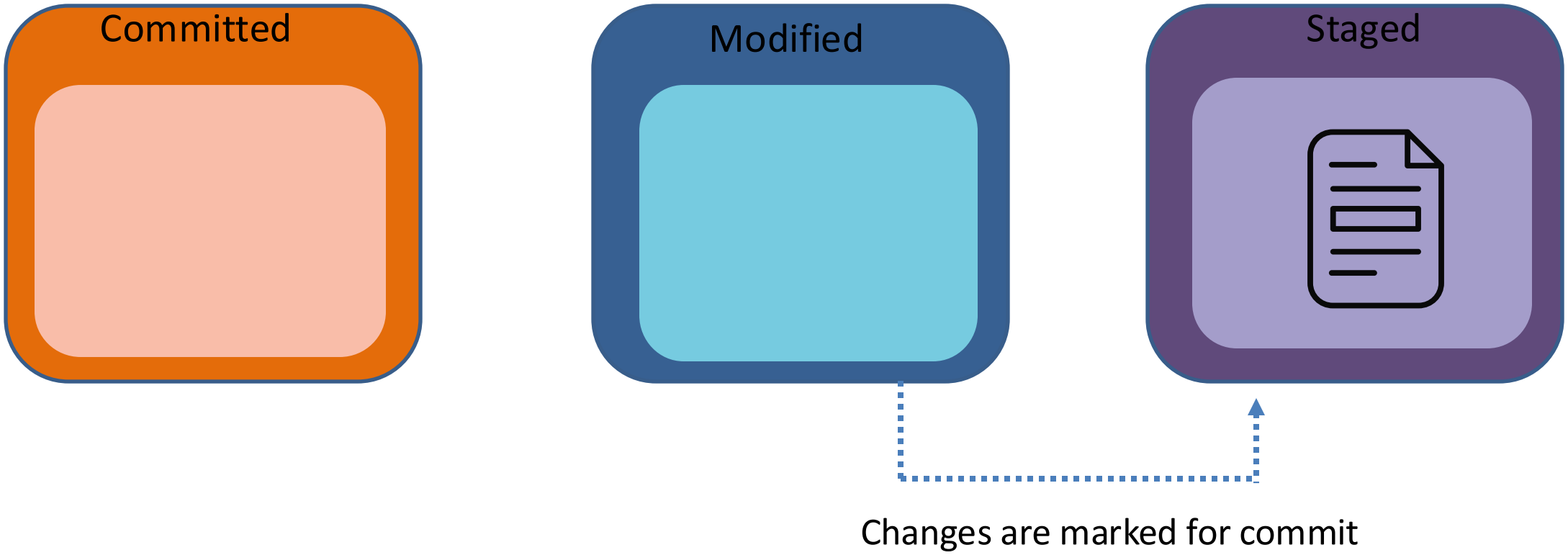
Three stages of a file



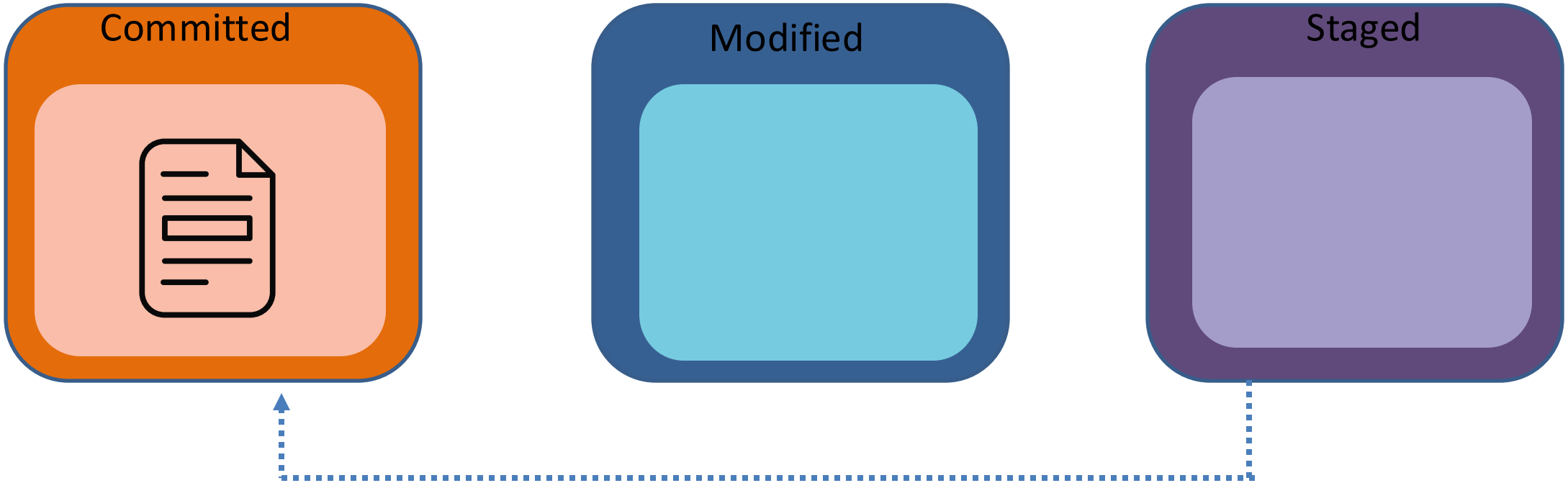
Three stages of a file



Three stages of a file

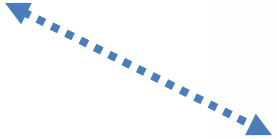


Three stages of a file



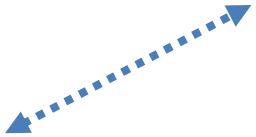
Git providers

GitHub

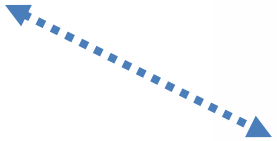


Git providers

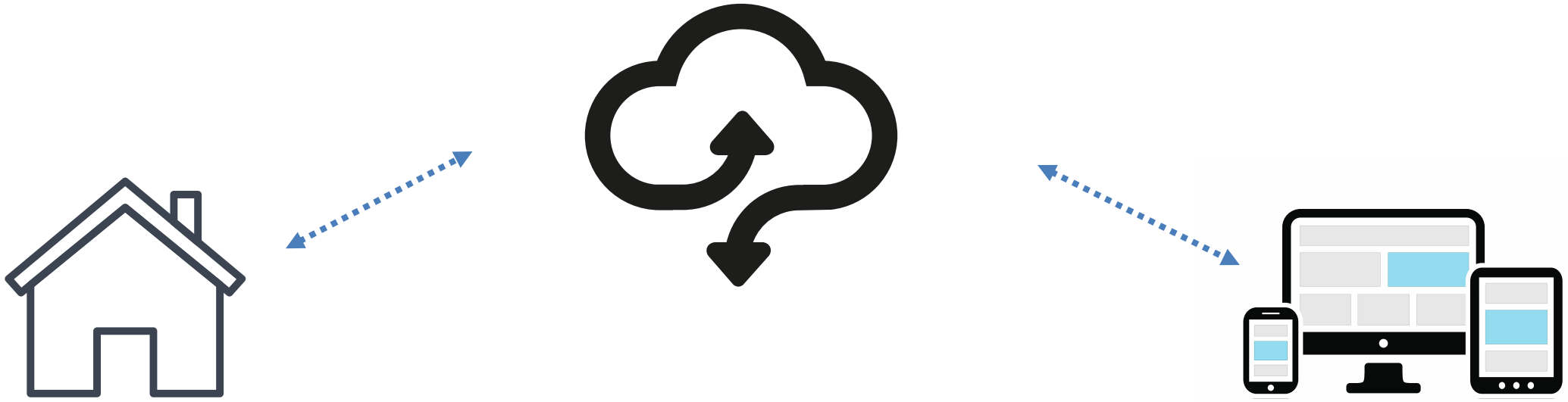
Git providers



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>...]
```

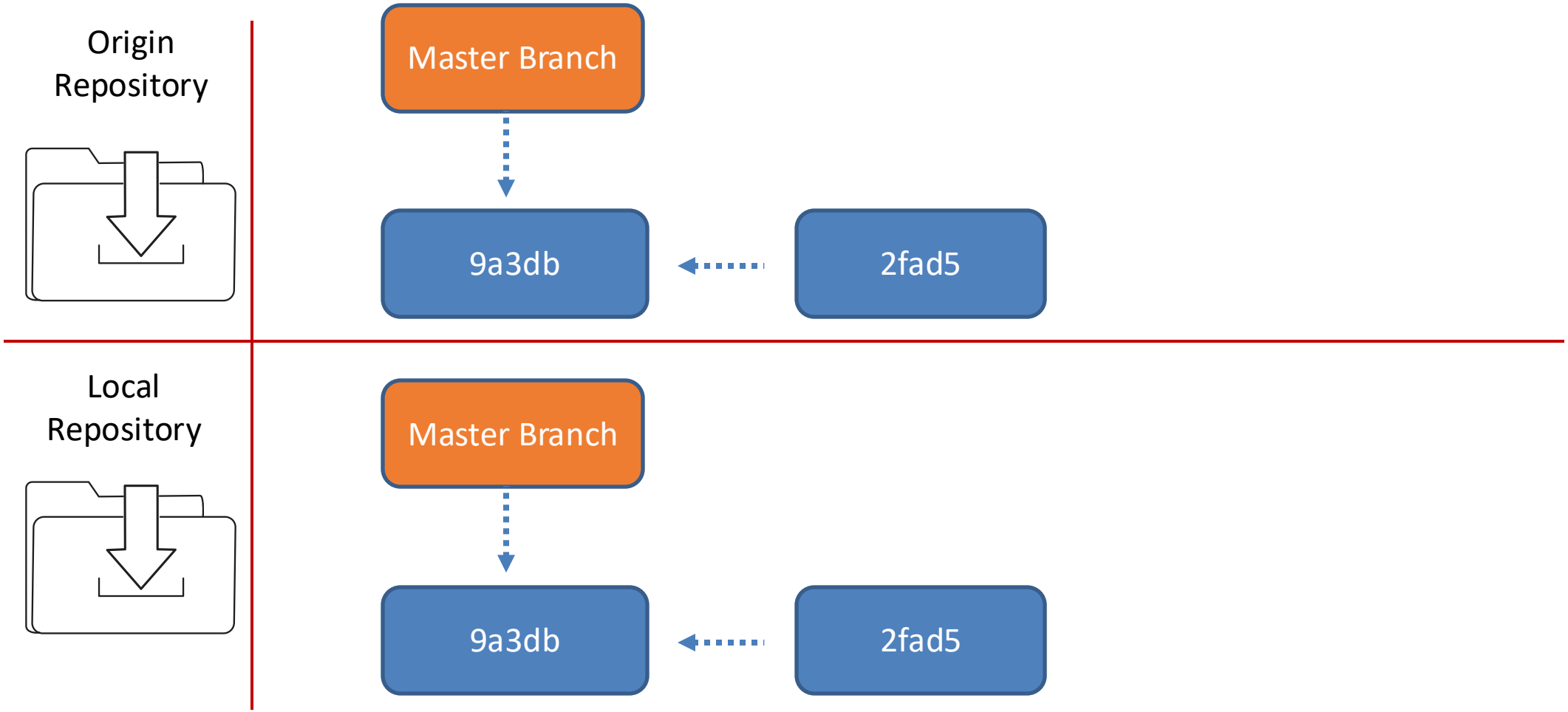
- `git status` displays the list of changed files together with the files that are yet to be staged or committed.

Git basic commands

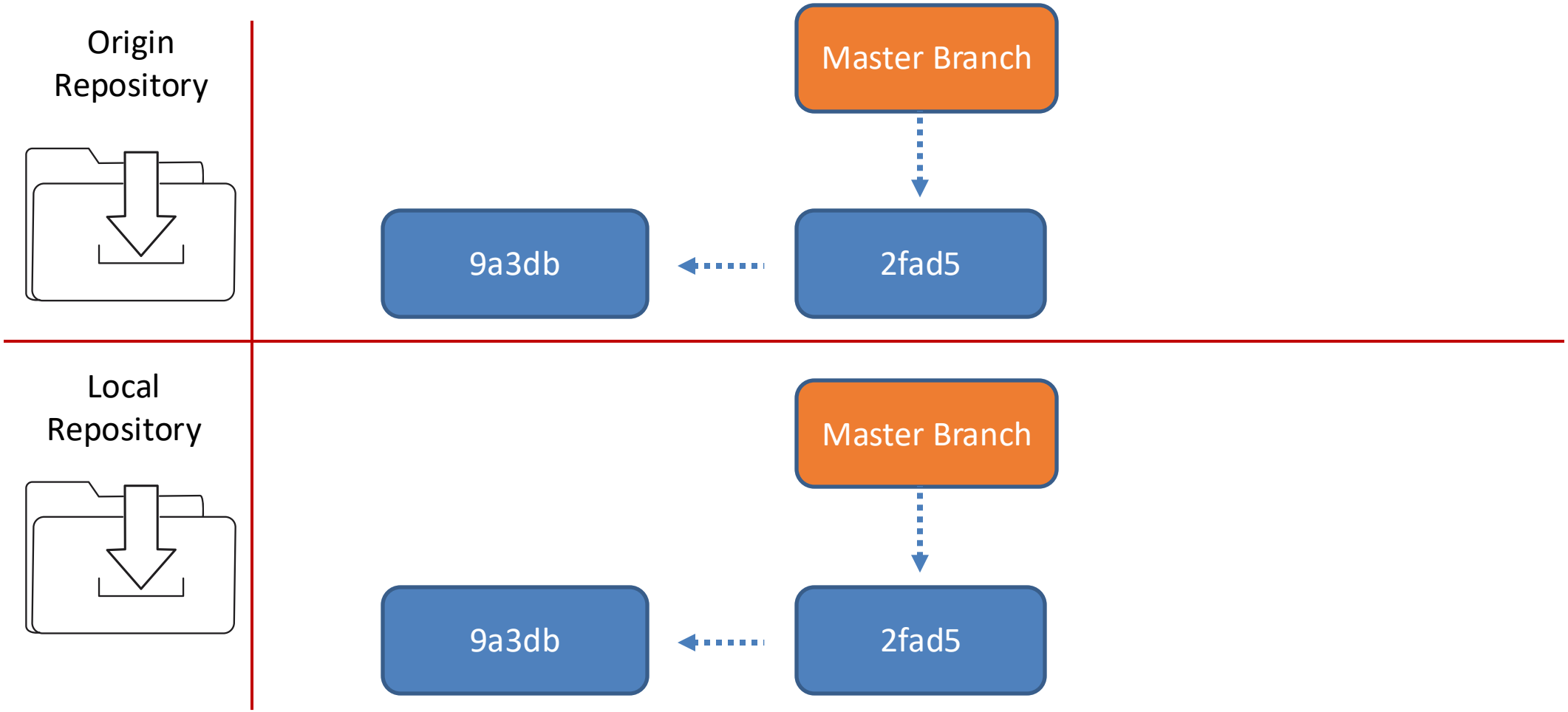
```
git push origin master
```

- `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 a68112e12f896ca028f25a46e58aaf46360cb847
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)
```

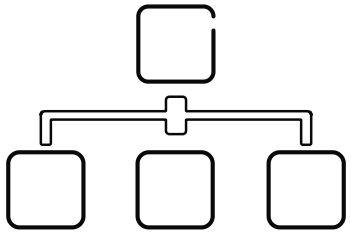
- `git reset` undo changes
- `--soft` `--mixed` or `--hard` depending on situation.
- `--soft` Does NOT change local working directory. Index still points to latest commit
- `--mixed` Does NOT change local working directory. Index points to reverted commit
- `--hard` Changes local working directory and 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?

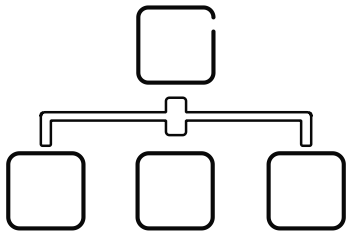
- Feature branches?
- All changes to master?

Software Development flows



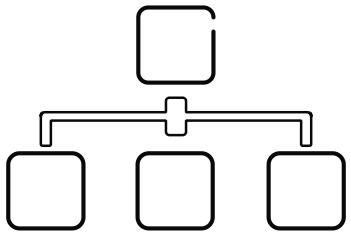
- Centralized flow
- GitHub
- Featured branch

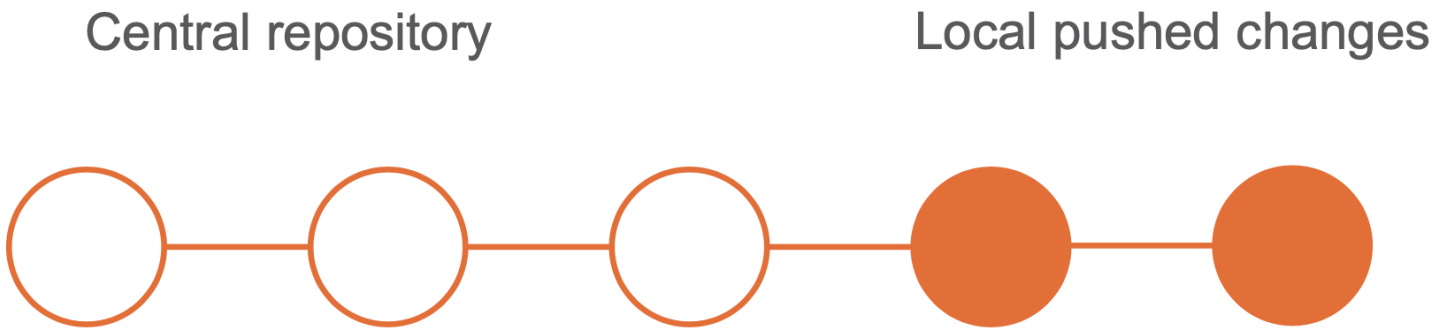
Centralized flow



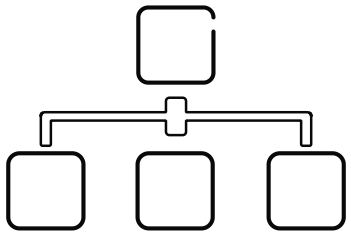
- Uses a centralized repository
- End users 'clone' repository
- Pull changes from central repository to local
- Make changes
- Push local changes back to centralized repository
- Similar to SVN

Centralized flow



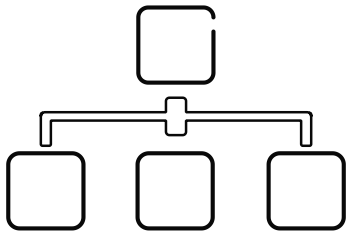


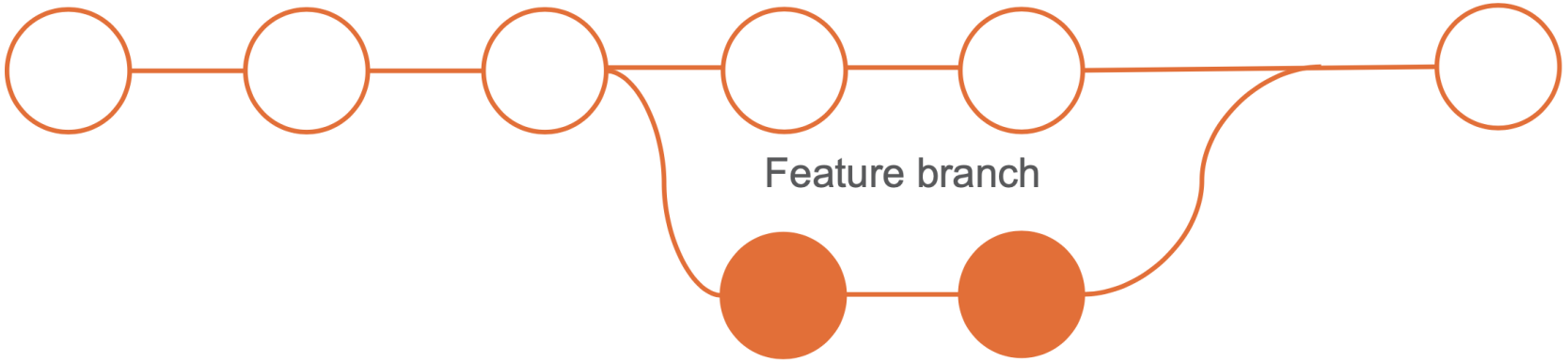
Feature branch flow



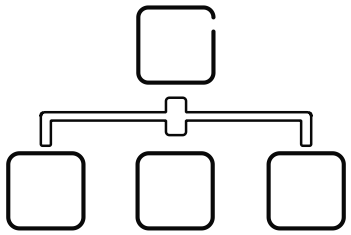
- Create a feature branch
- Add new code
- Open a pull request
- Merge code back into Master

Feature branch flow

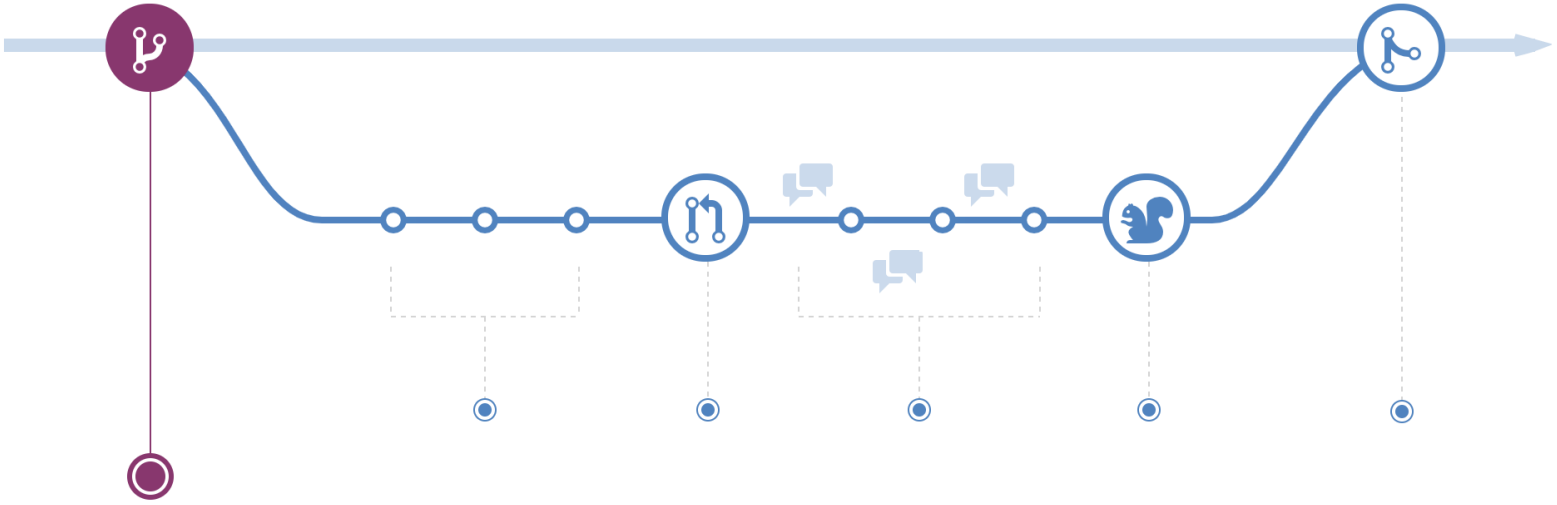




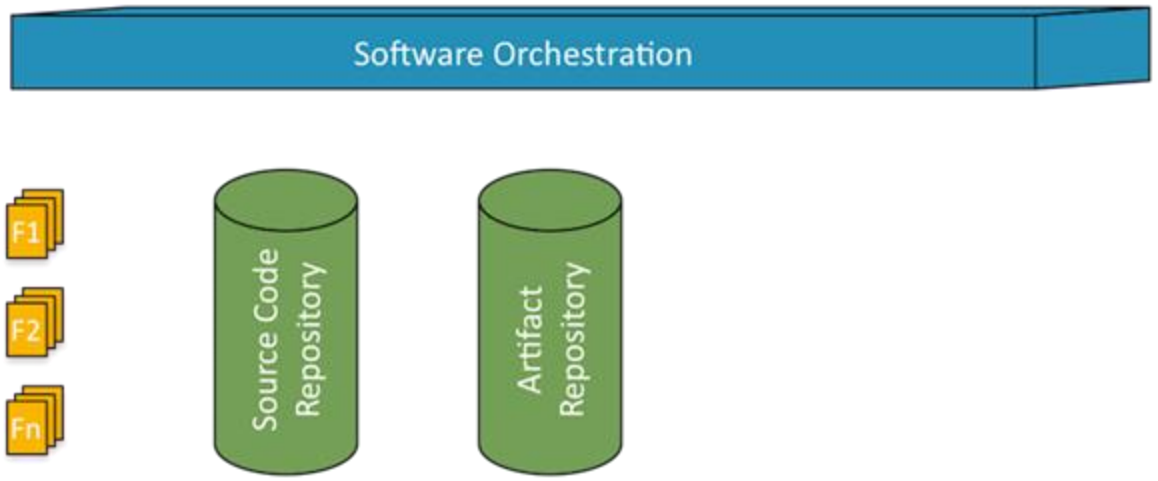
GitHub flow



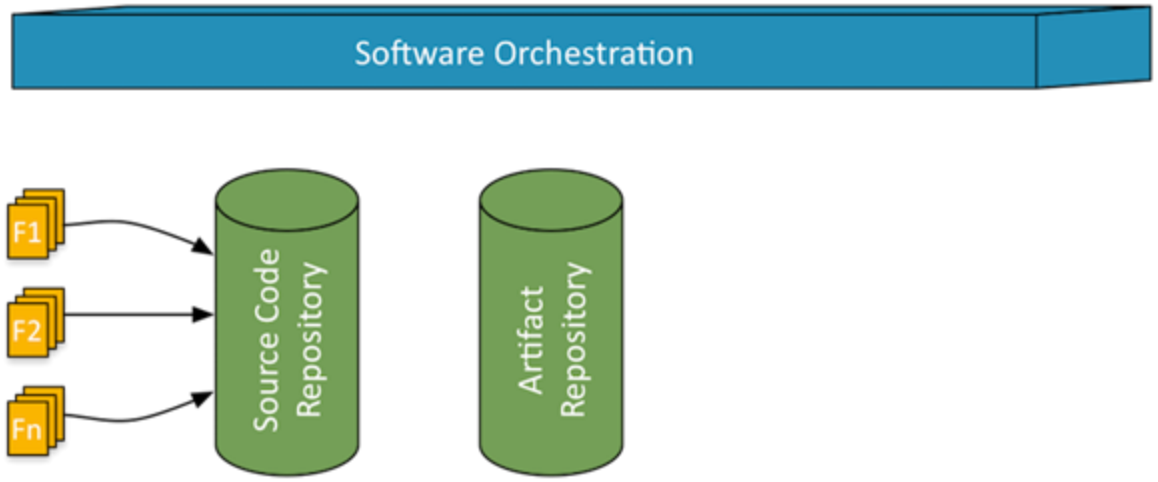
1. Create a branch



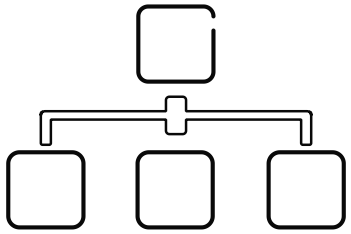
GitHub flow



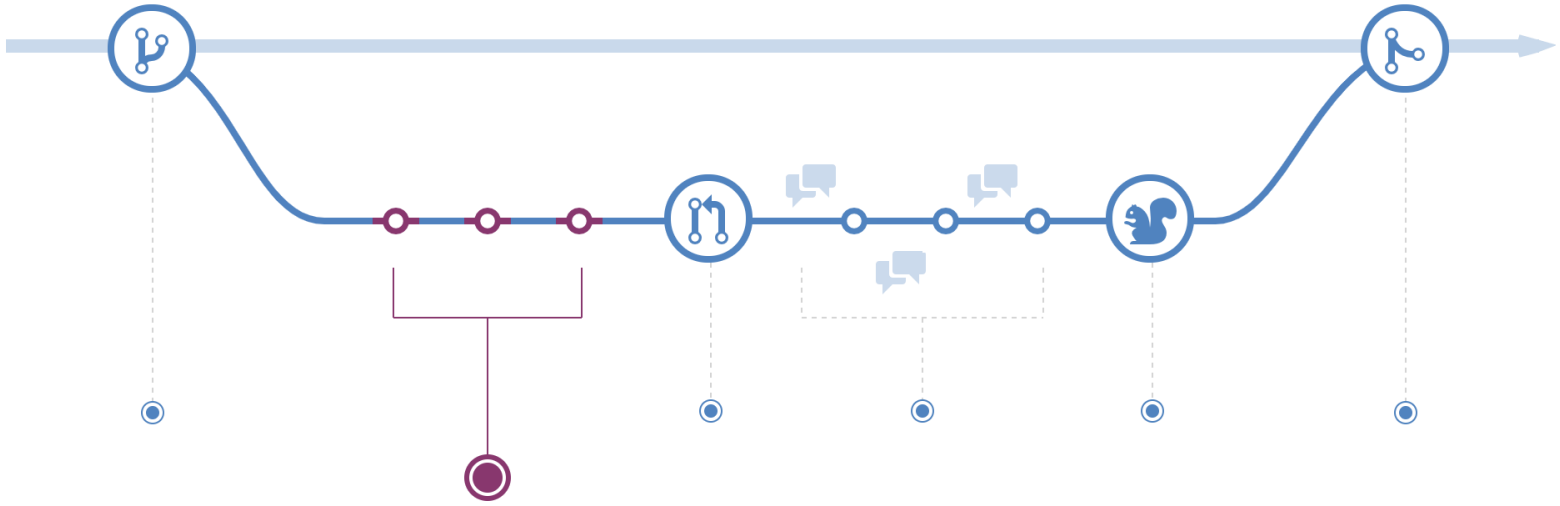
GitHub flow



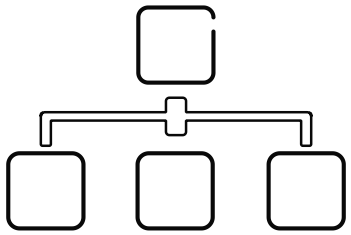
GitHub flow



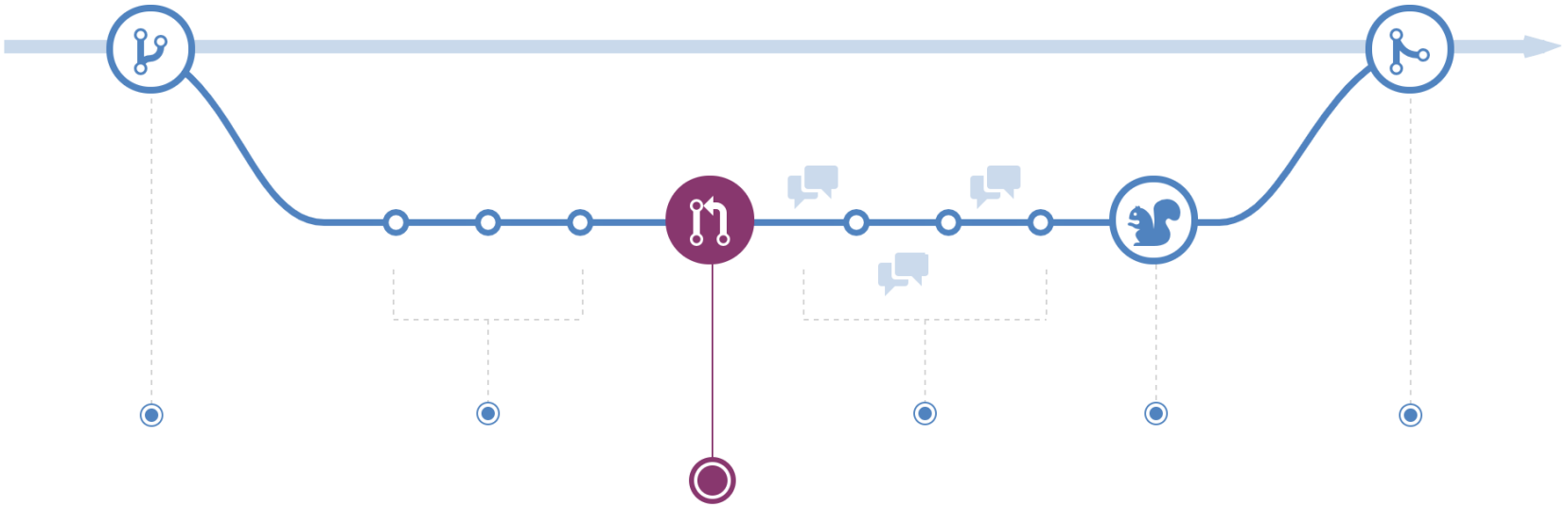
1. Create a branch
2. Checkout branch and make changes



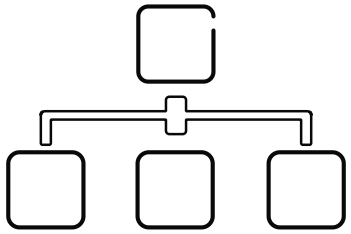
GitHub flow



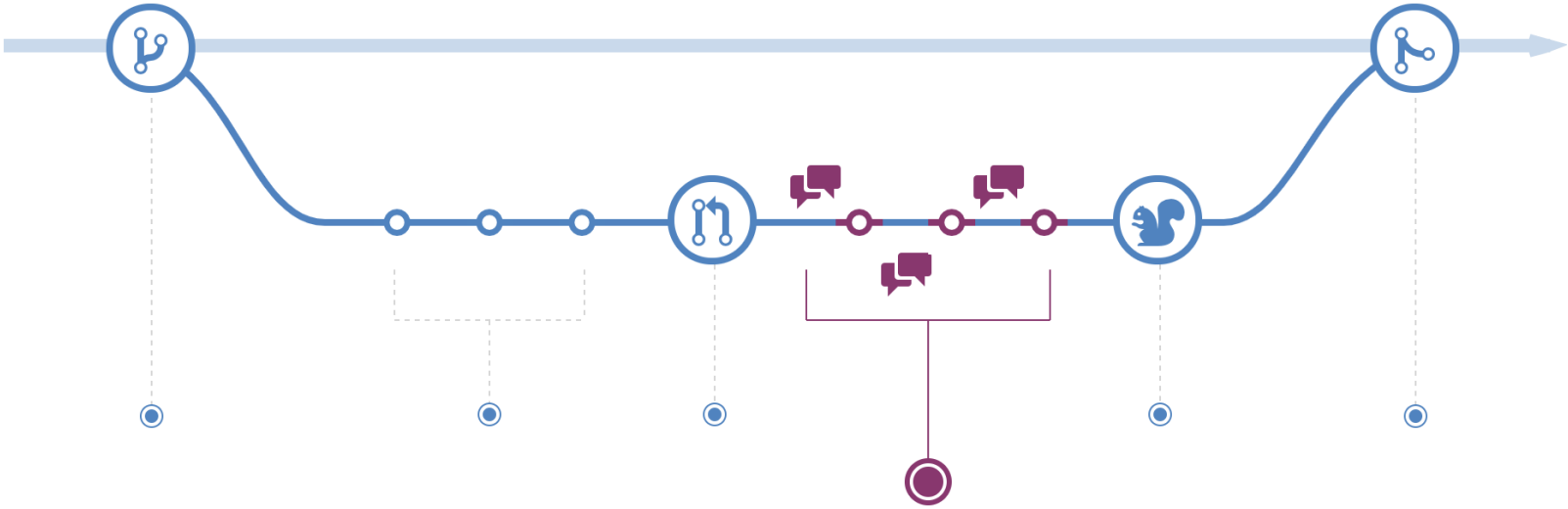
1. Create a branch
2. Checkout branch and make changes
3. Open pull request



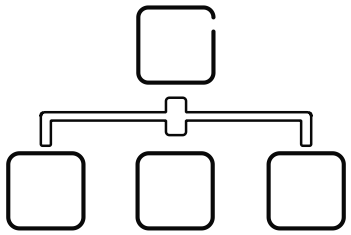
GitHub flow



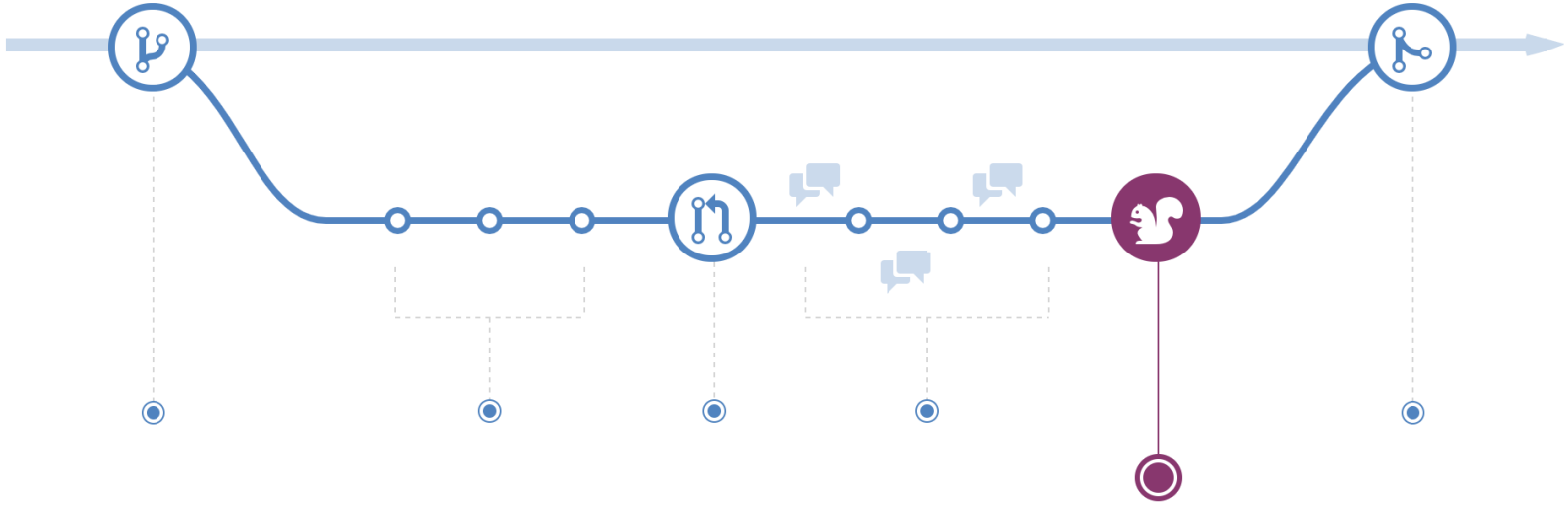
1. Create a branch
2. Checkout branch and make changes
3. Open pull request
4. Peer review



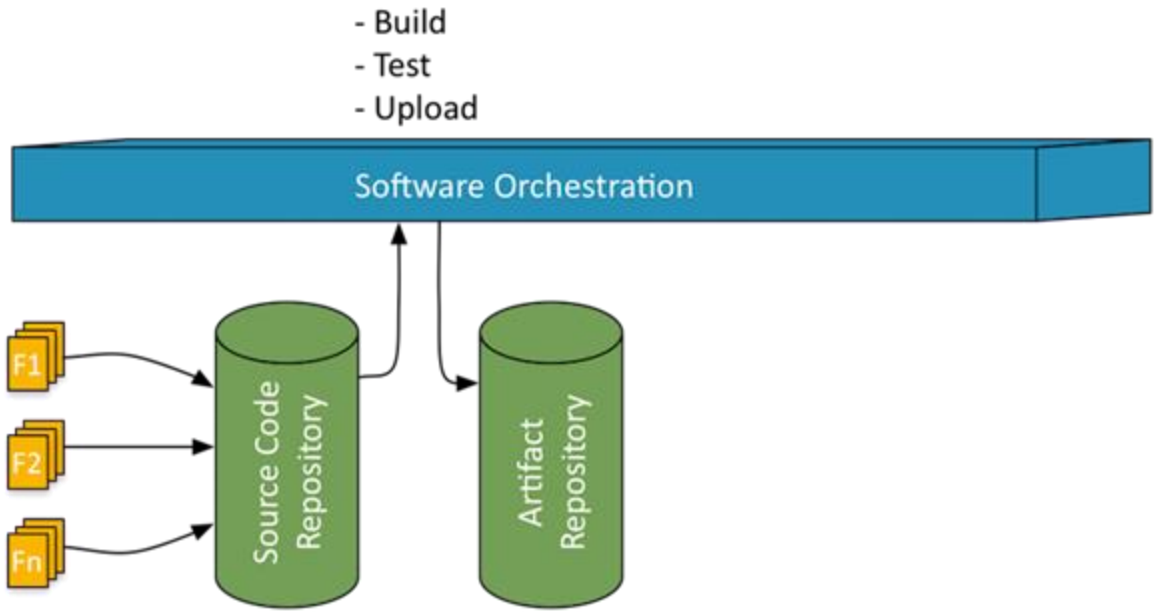
GitHub flow



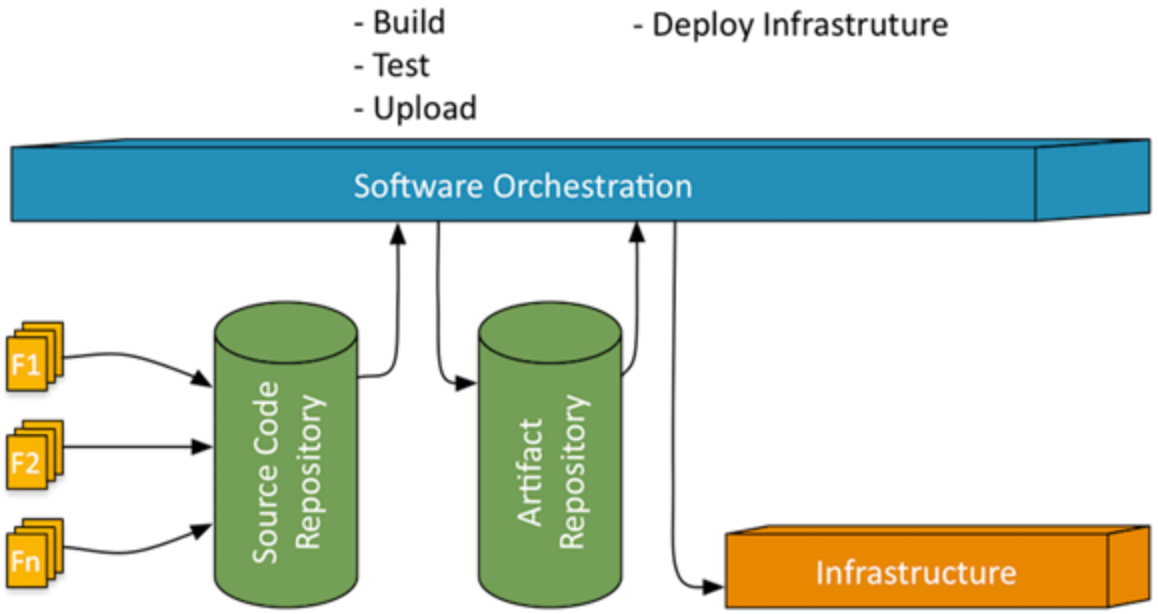
1. Create a branch
2. Checkout branch and make changes
3. Open pull request
4. Peer review
5. Deploy



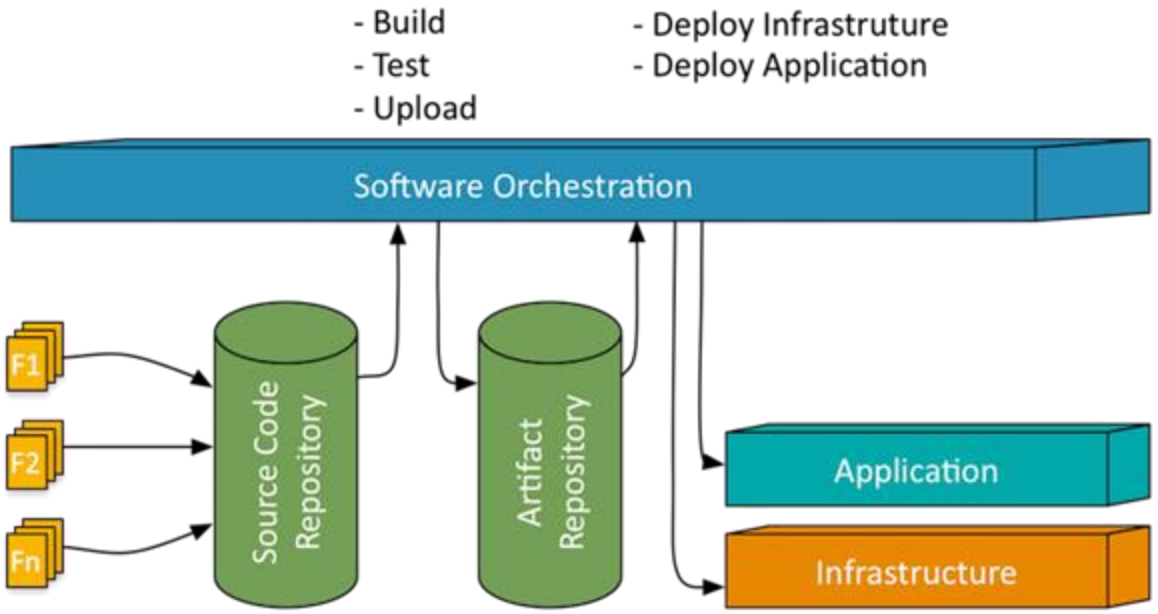
Continuous Integration & Testing



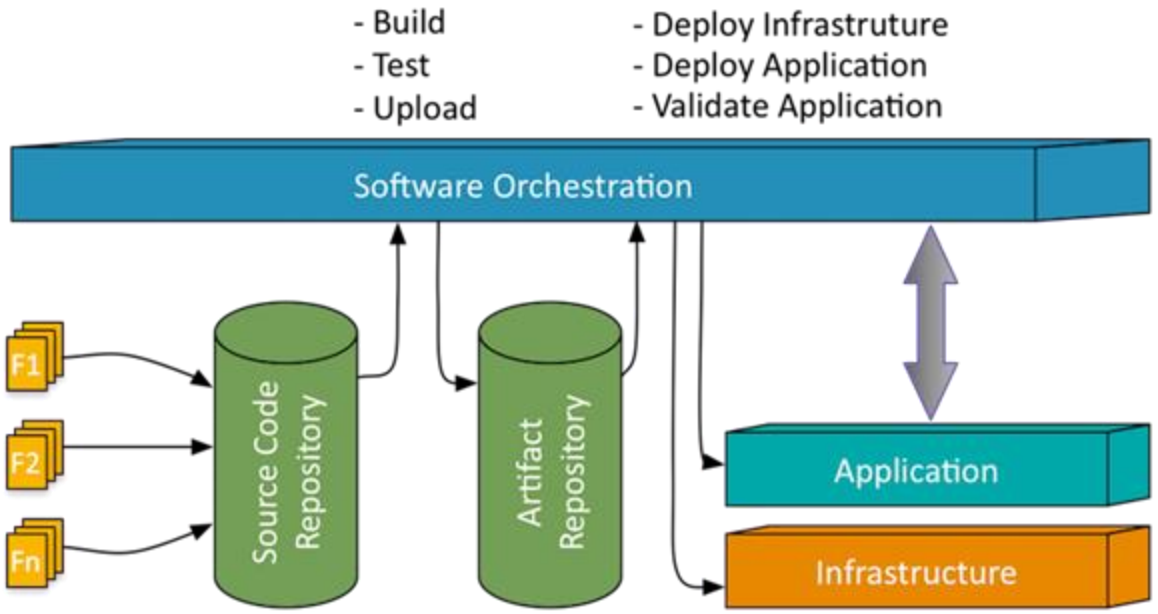
Continuous Release and Deployment



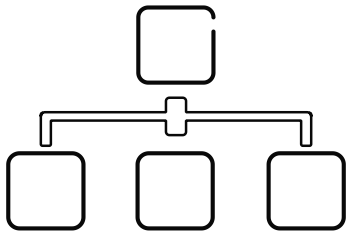
Continuous Release and Deployment



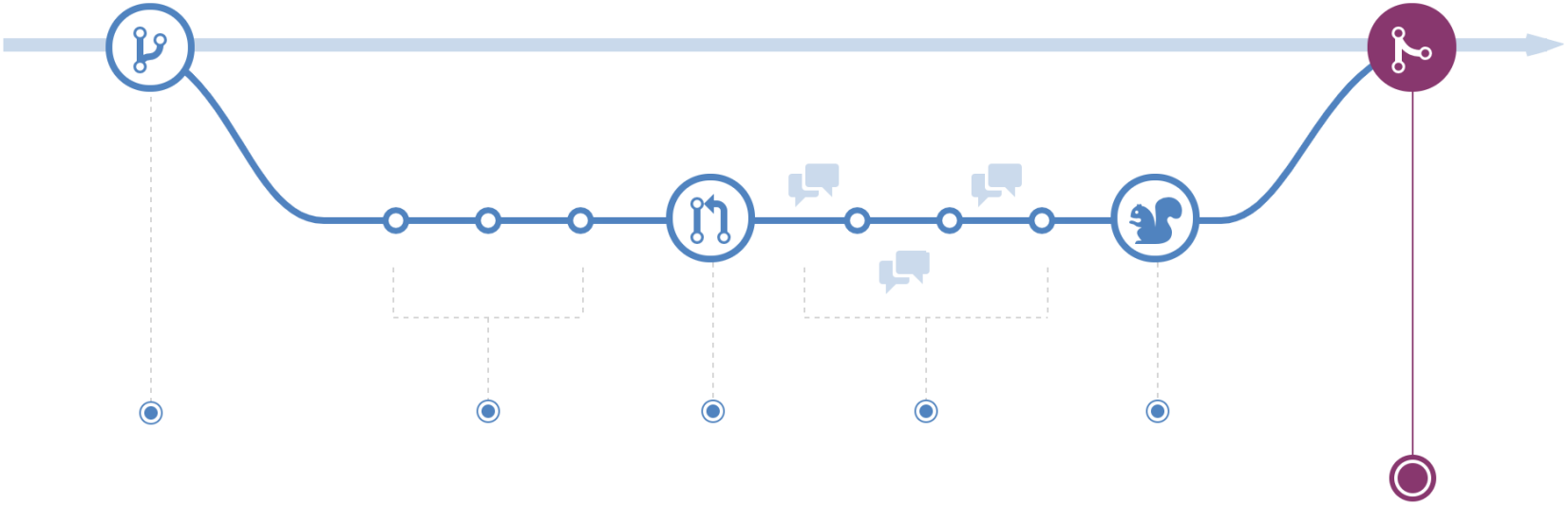
Continuous Release and Deployment



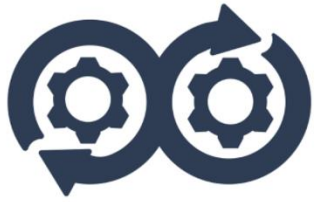
GitHub flow



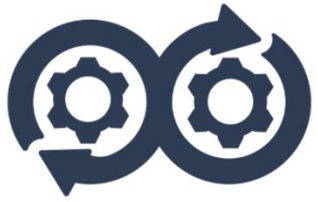
1. Create a branch
2. Checkout branch and make changes
3. Open pull request
4. Peer review
5. Deploy
6. Merge into Master branch



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:

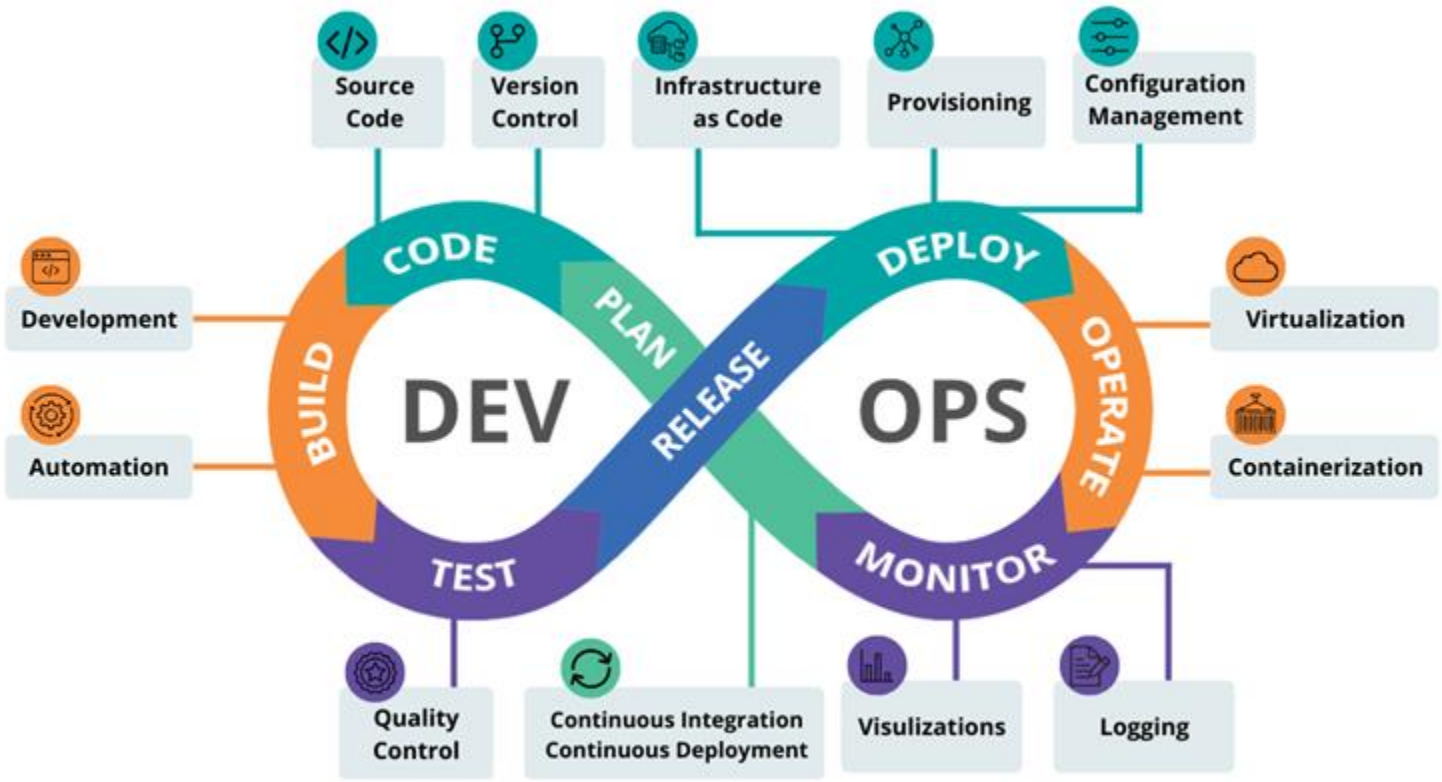
- Operating System
- Runtime
- Libraries
- Utilities
- Configuration files

- 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 investinvestigation of testing solutions.

Azure Repos

Provides full-support for cloud-hosted private repositories.

Azure Artifacts

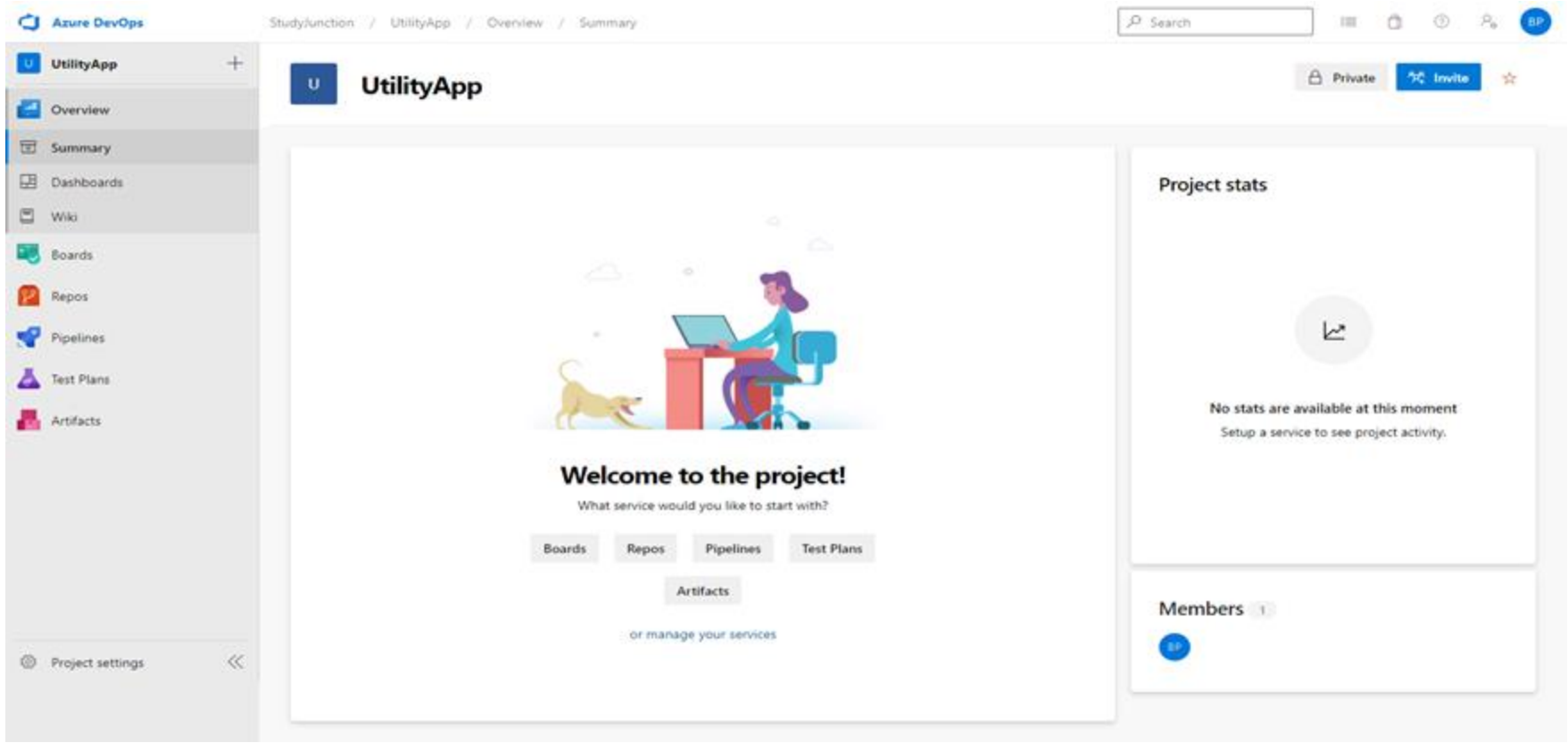
Package management Support for Maven, npm, NuGet and Python package feeds from private or public sources.

Azure Pipelines

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

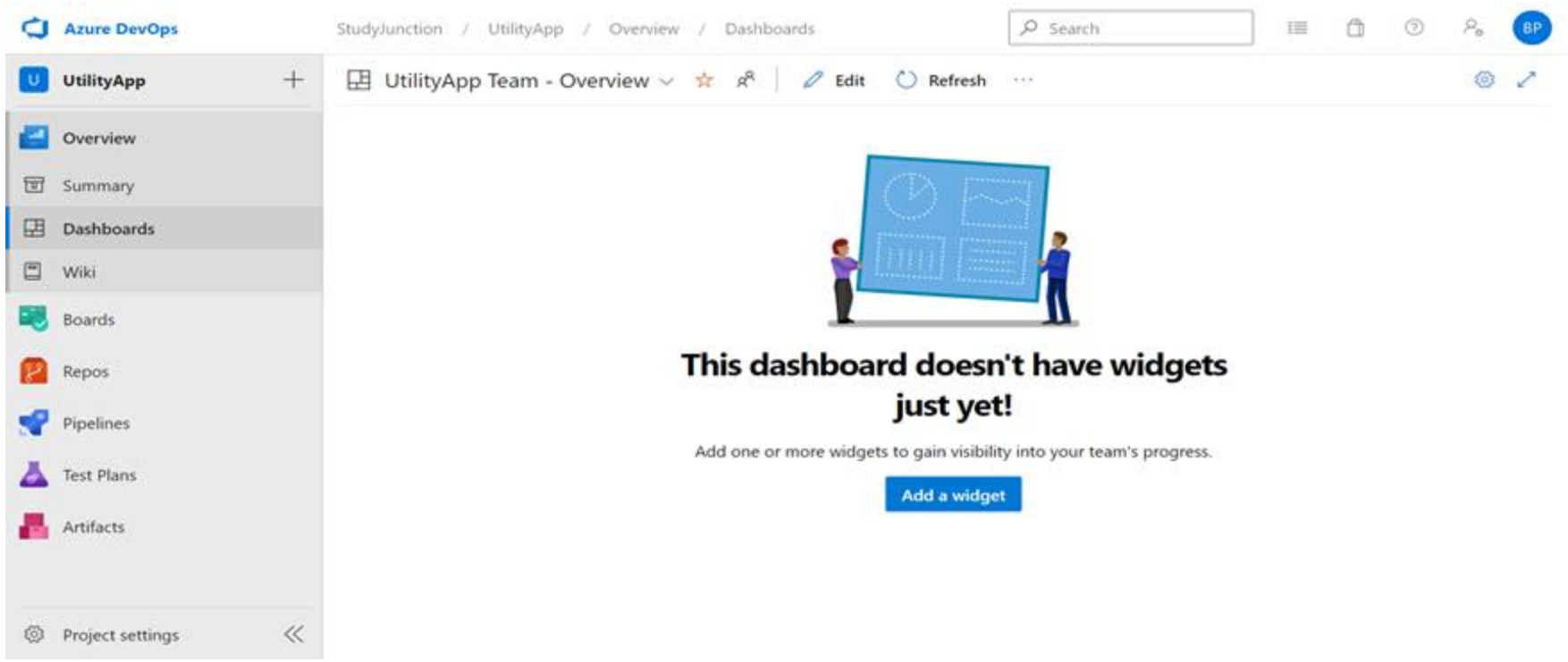
Overview | Summary

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-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.



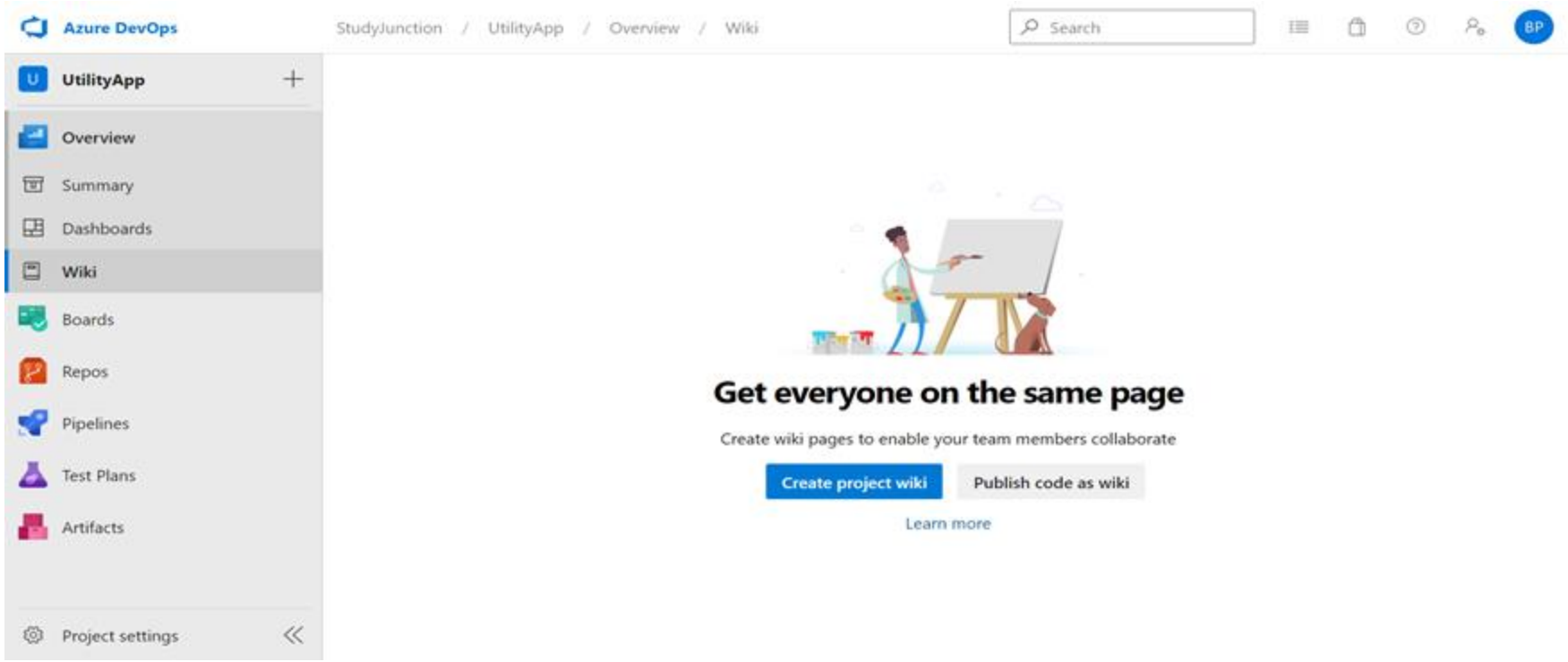
Overview | Dashboards

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.



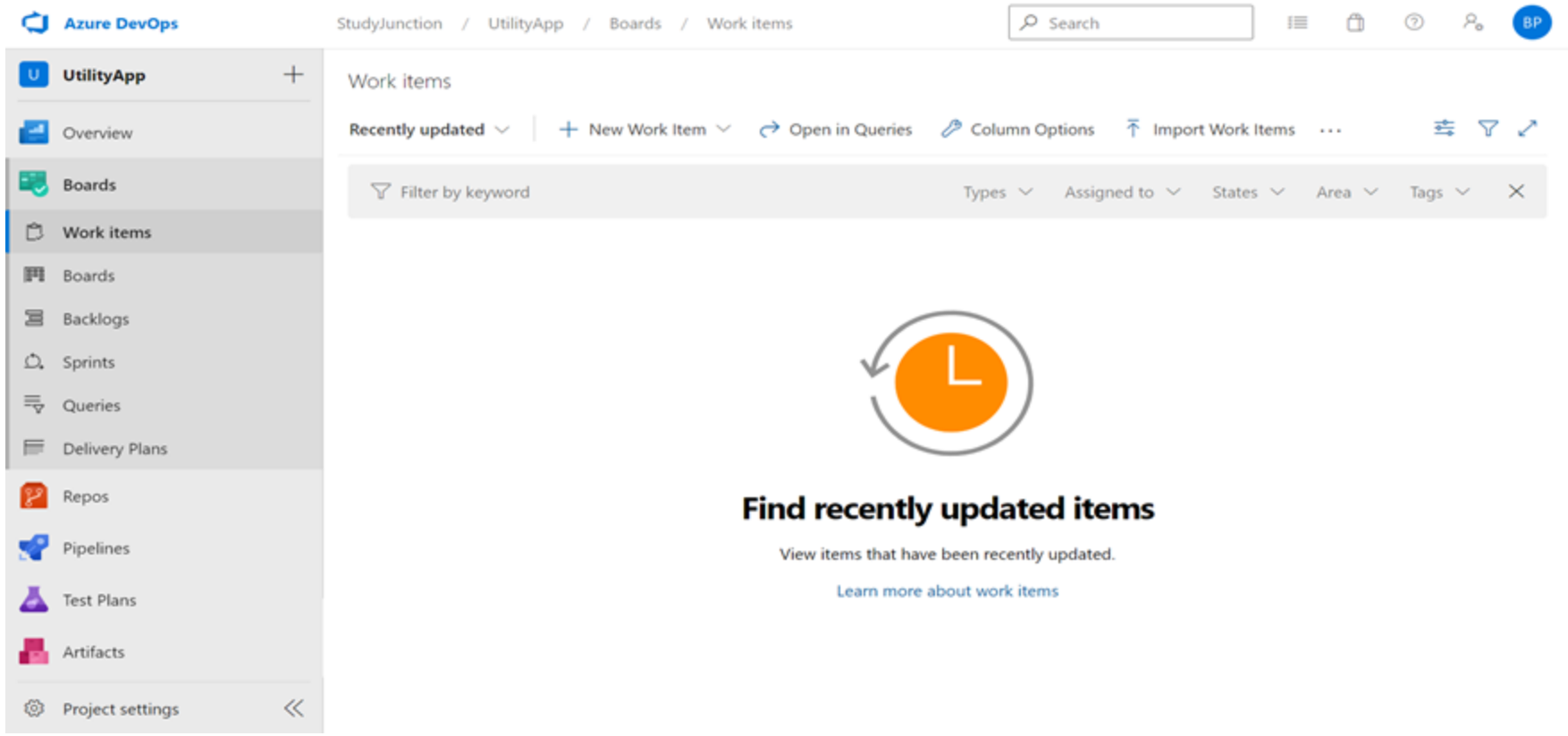
Overview | Wiki

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.



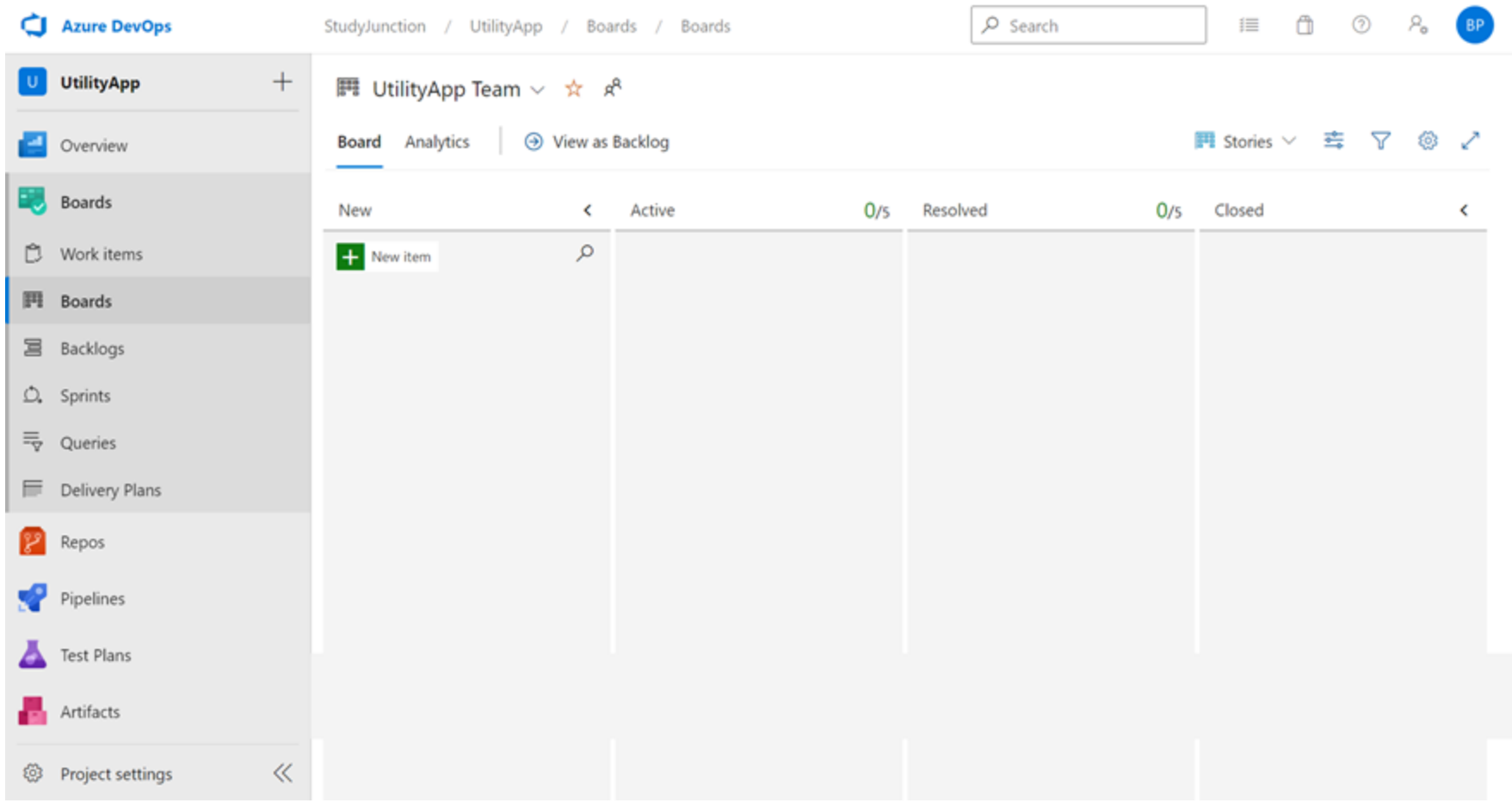
Boards | Work items

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



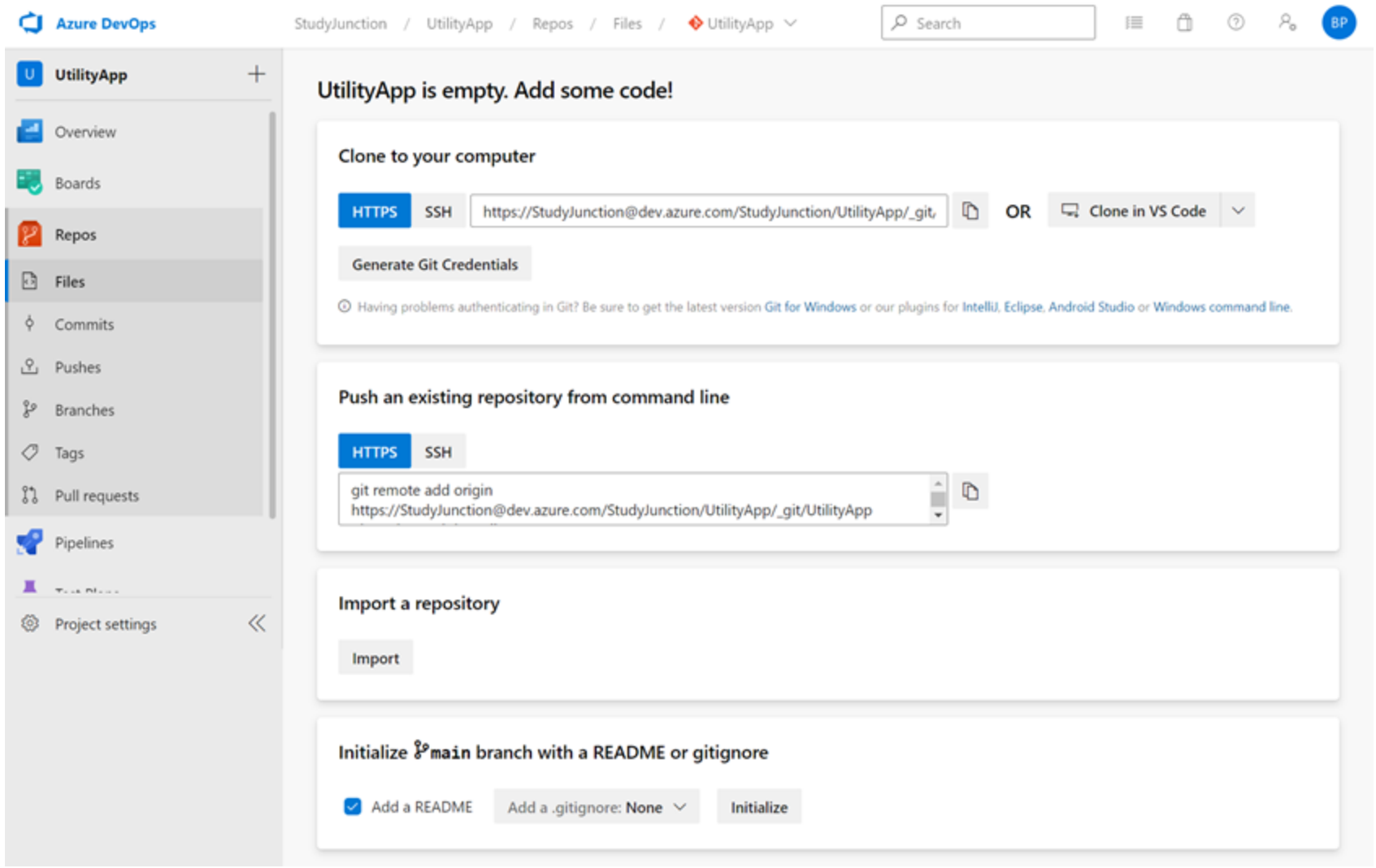
Boards | Boards

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.



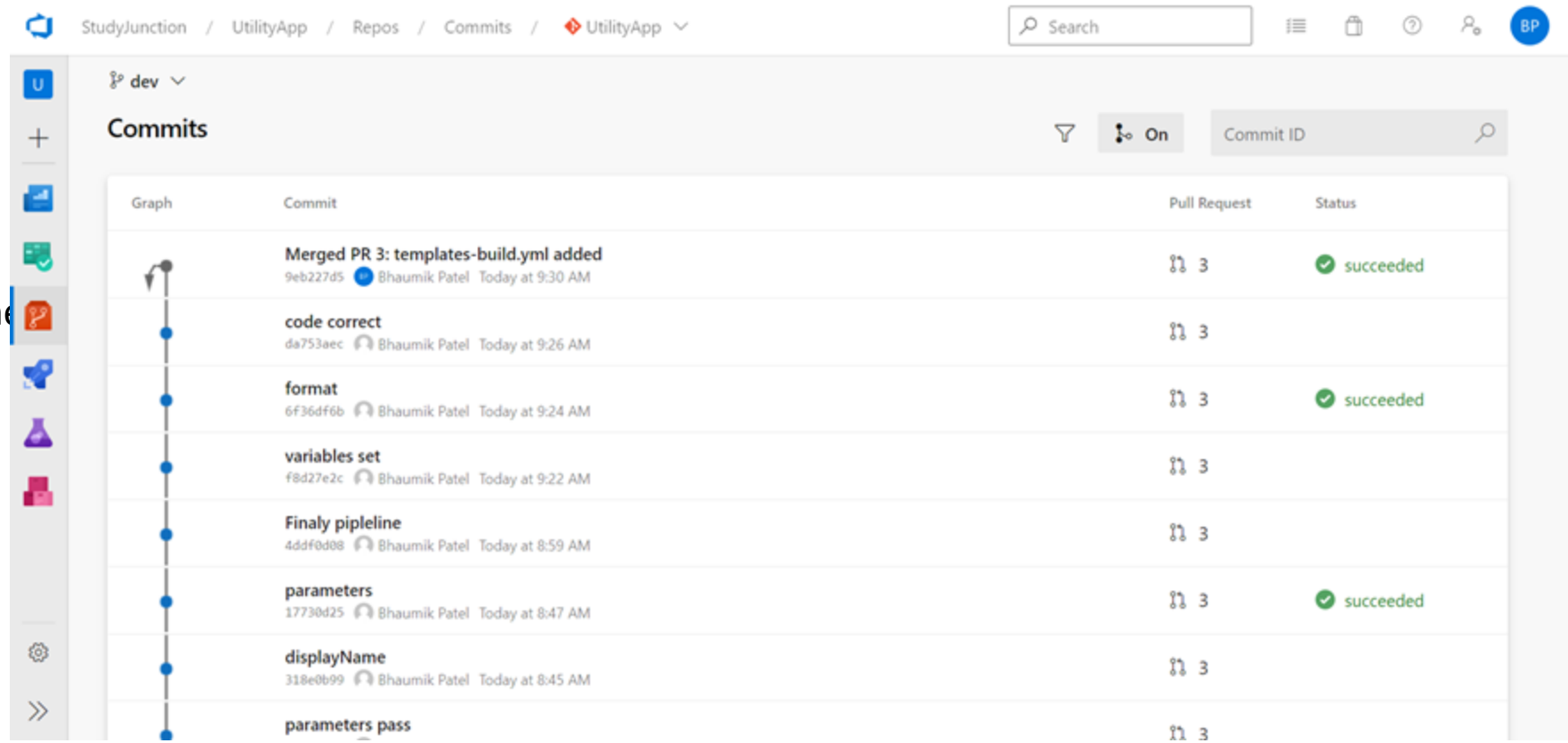
Repos | Files

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.



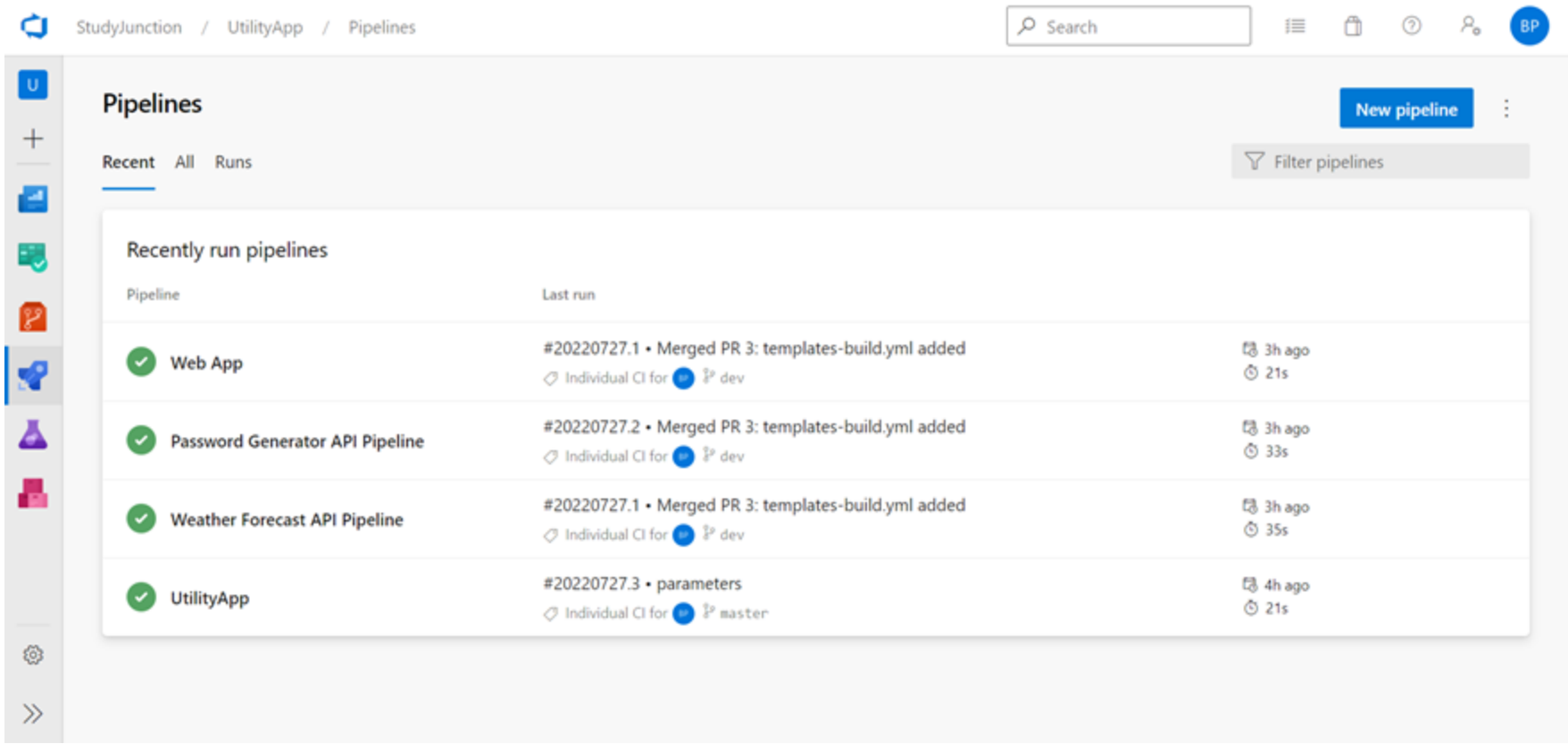
Repos | Commits

The Commits page lists all the
recover one specific version.
commit timeline.



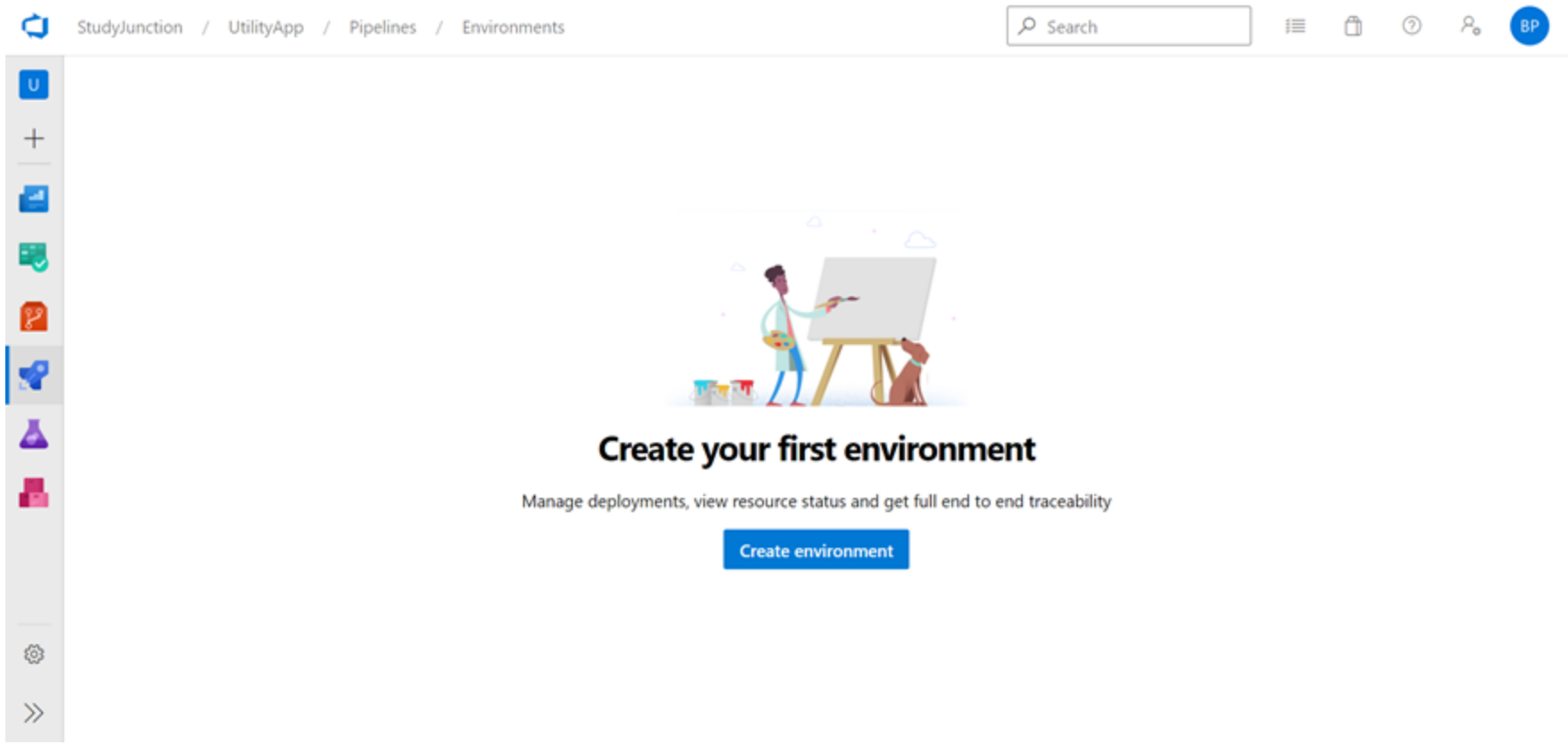
Pipelines | Pipelines

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



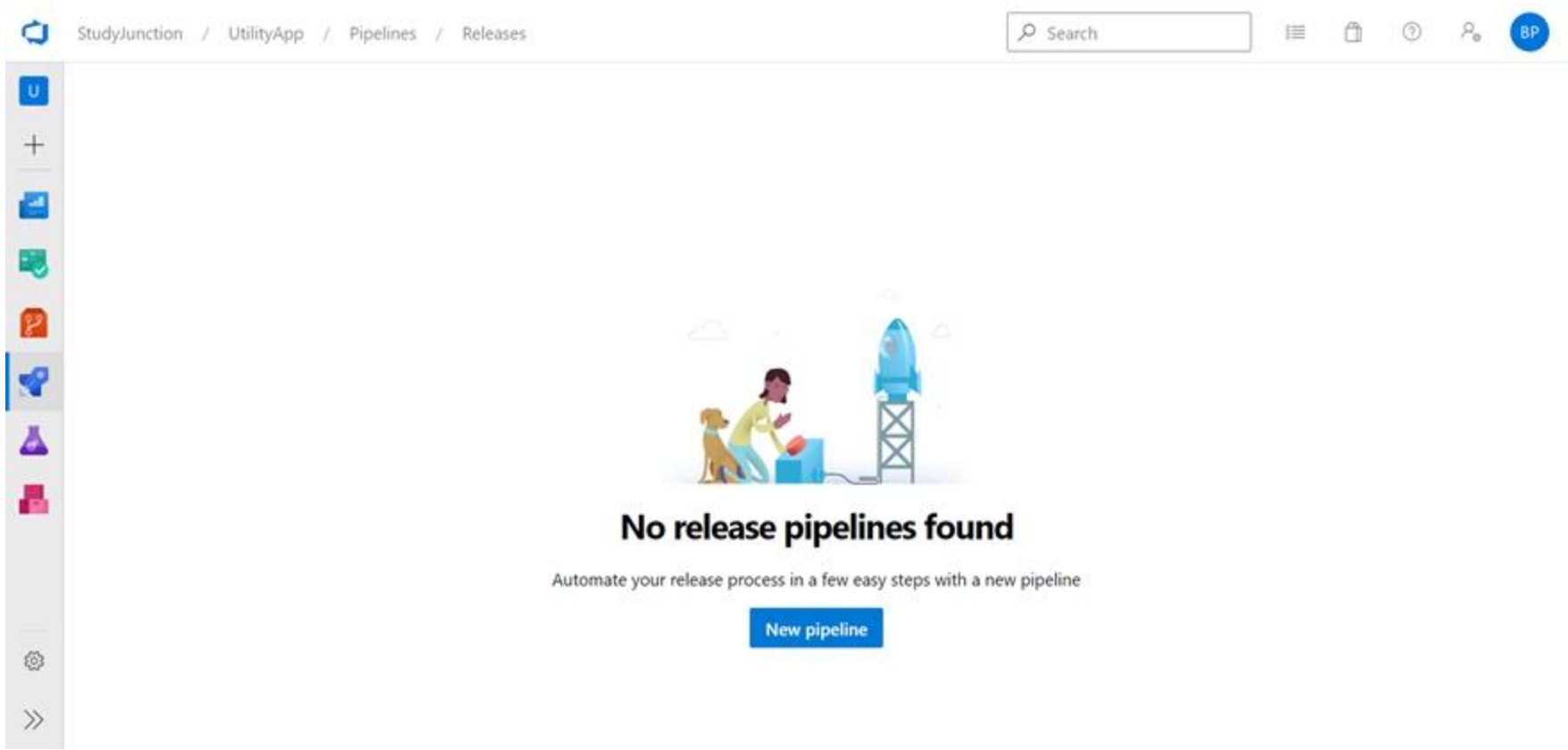
Pipelines | Environment

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



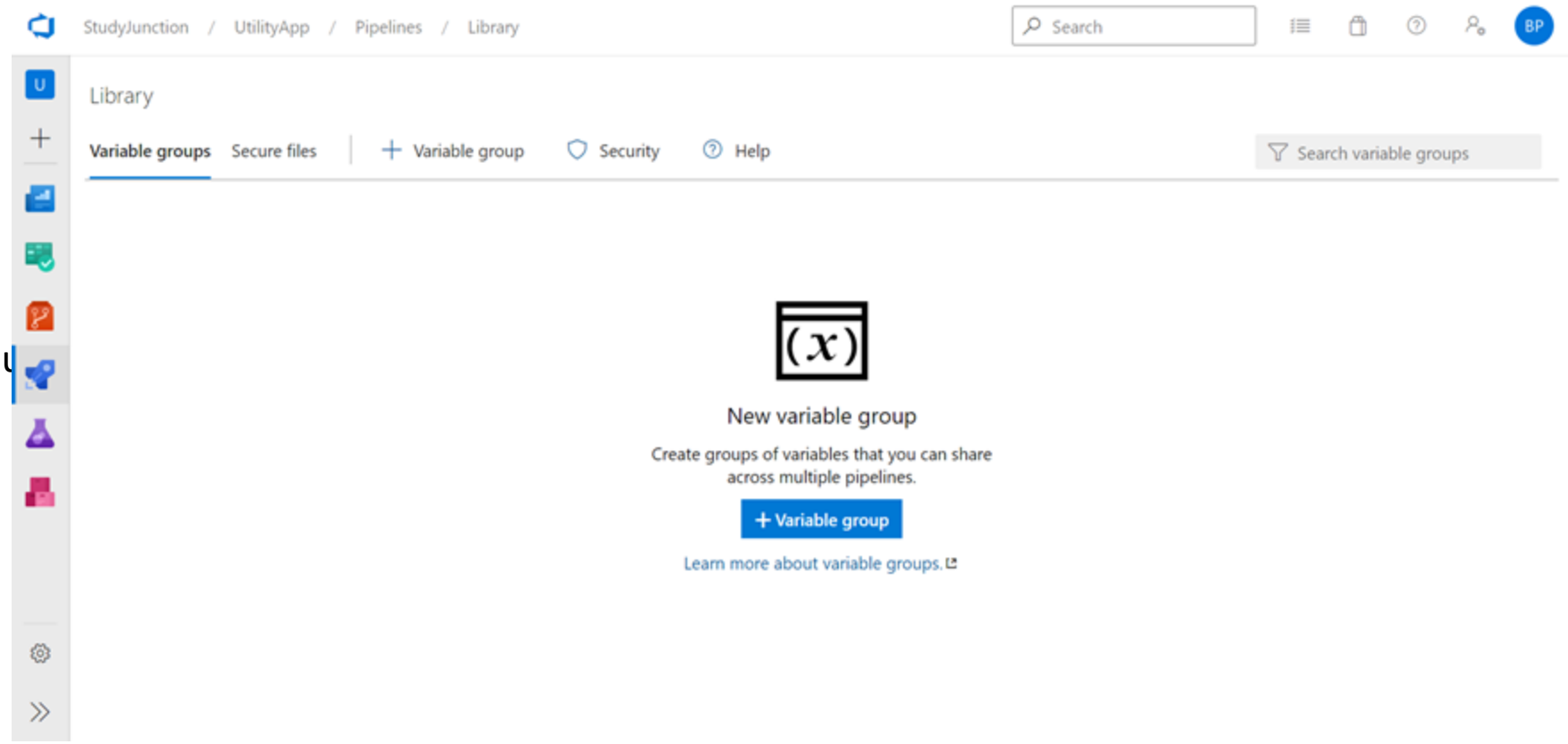
Pipelines | Releases

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.



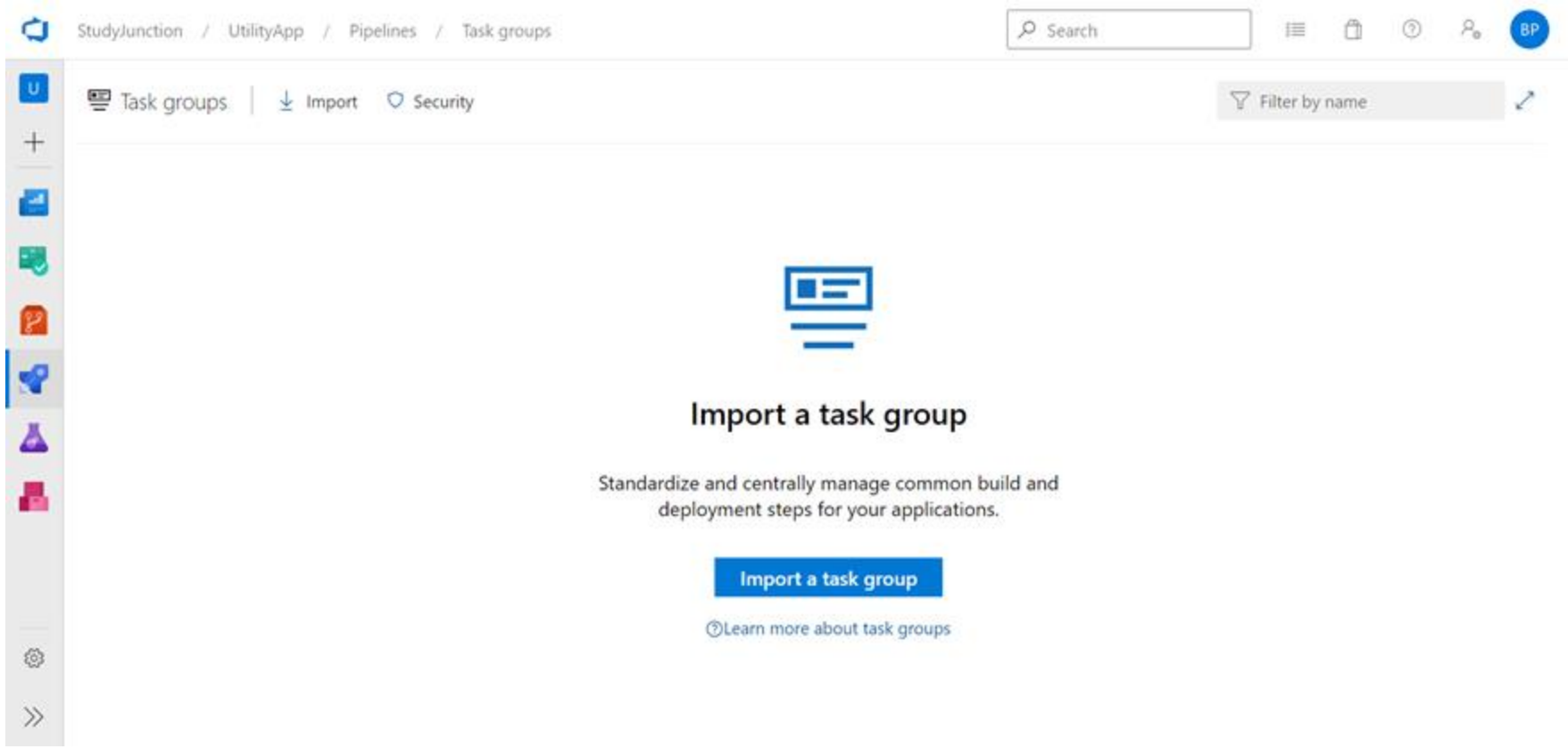
Pipelines | Library

Variable groups store values across multiple pipelines. You



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.



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.