What it takes to be a DevOps engineer ?

~ whoami

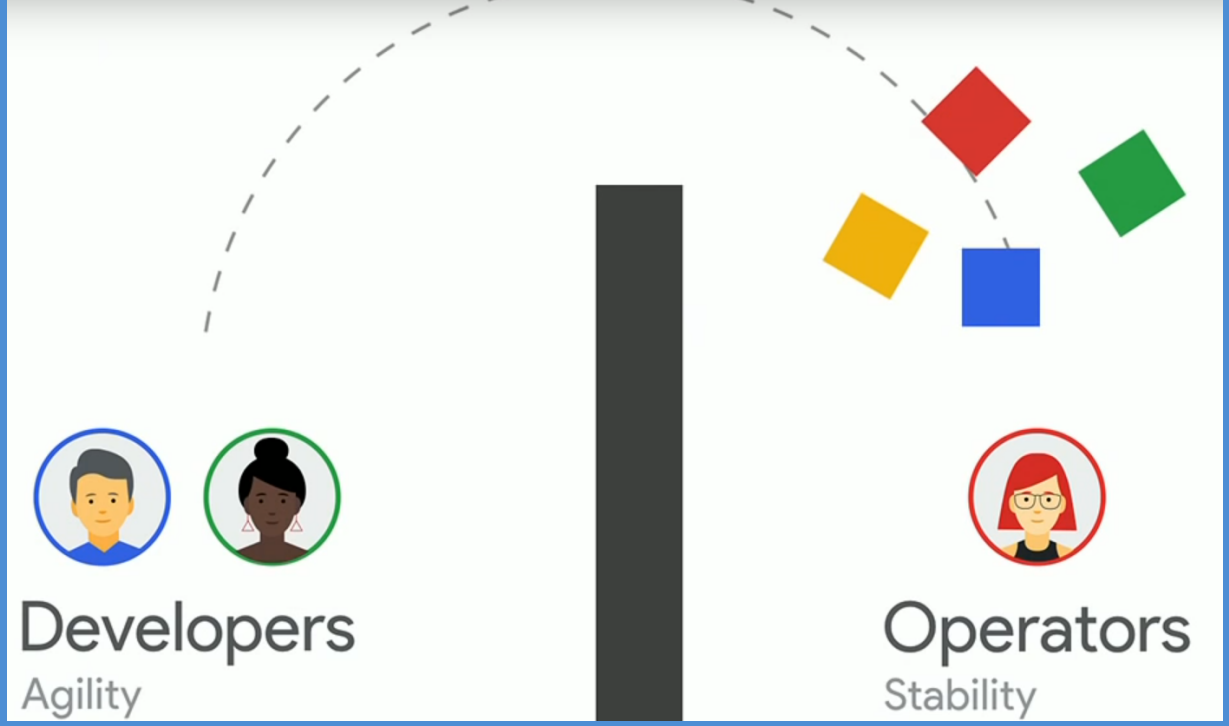
- I'm Madhan
- Working as a DevSecOps engineer at TrusTrace
- Having 8+ years of experience
- I also write blogs, with a total of 10+ posts and 40k+ views to date.
- And I maintain few interesting projects at Github
- You can connect with me on Linkedin

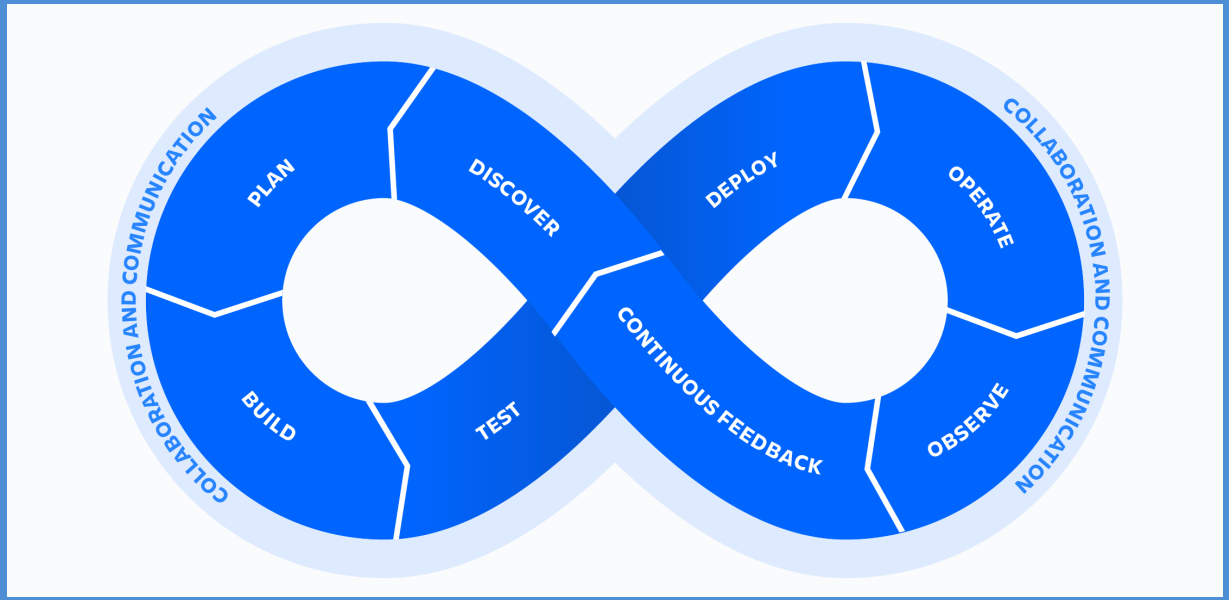
Is DevOps a Job title or Culture ?

* ***Not here to start a flame war***

What is DevOps ?

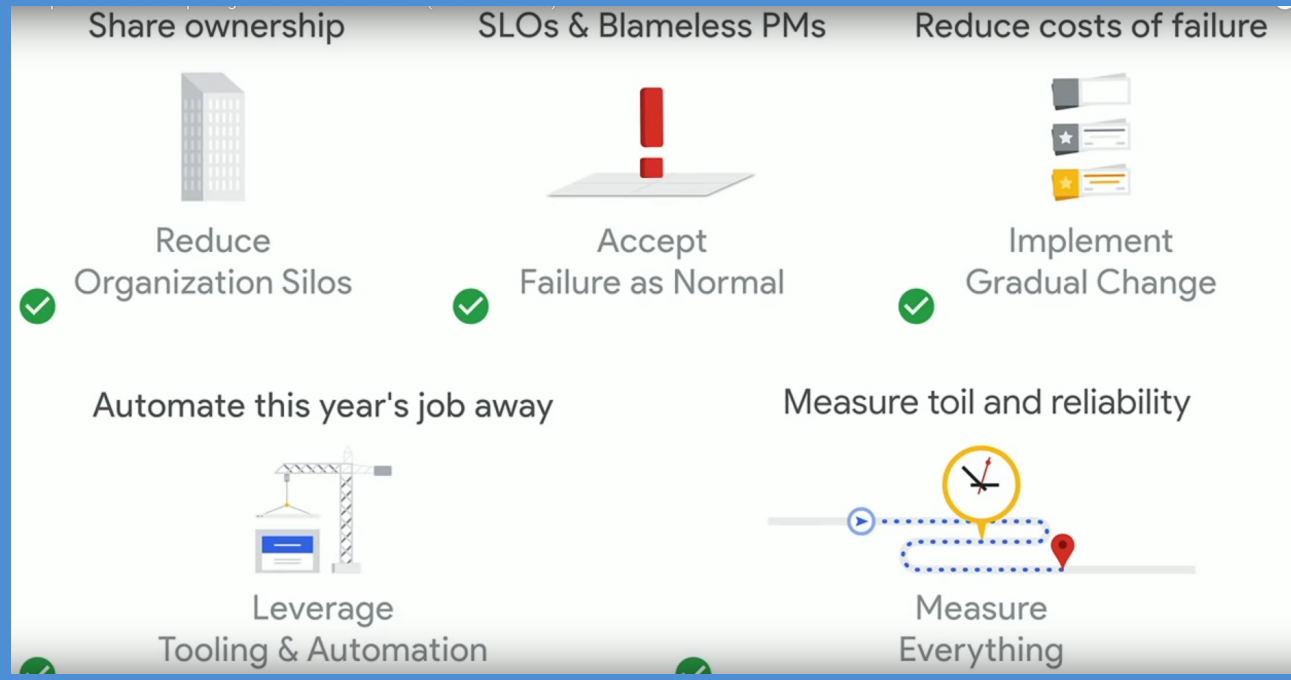
DevOps is a set of practices, tools, and a cultural philosophy that automate and integrate the processes between software development and IT teams.





How does it work ?

- Share ownership
- SLOs & Blameless PMs
- Reduce cost of failure
- Automate
- Measure toil and reliability



Class SRE implements DevOps

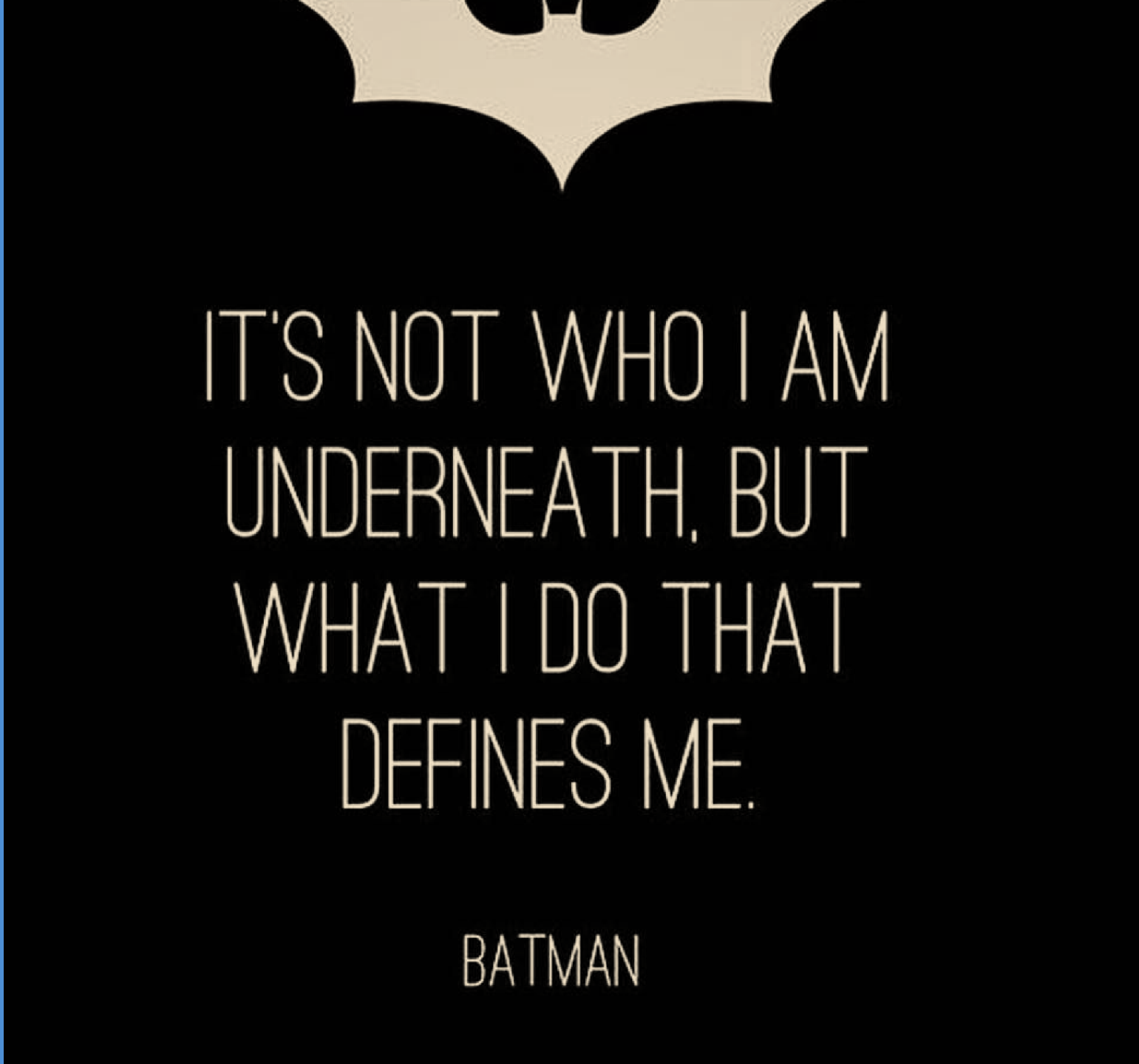
You build it, you run it

Operate What You Build

Tools, Principles and Workflow are aligned with the Culture, People, Team and Organization. And it's very essential it can either make or break the product

Then why your job title says DevOps ?

*So should I be called **SRE**, **Infrastructure Engineer**,**Platform engineer**, **Release Engineer** or **Cloud engineer**.*



First principle thinking

A first principle is a basic proposition or assumption that cannot be deduced from any other proposition or assumption

@addyosmani

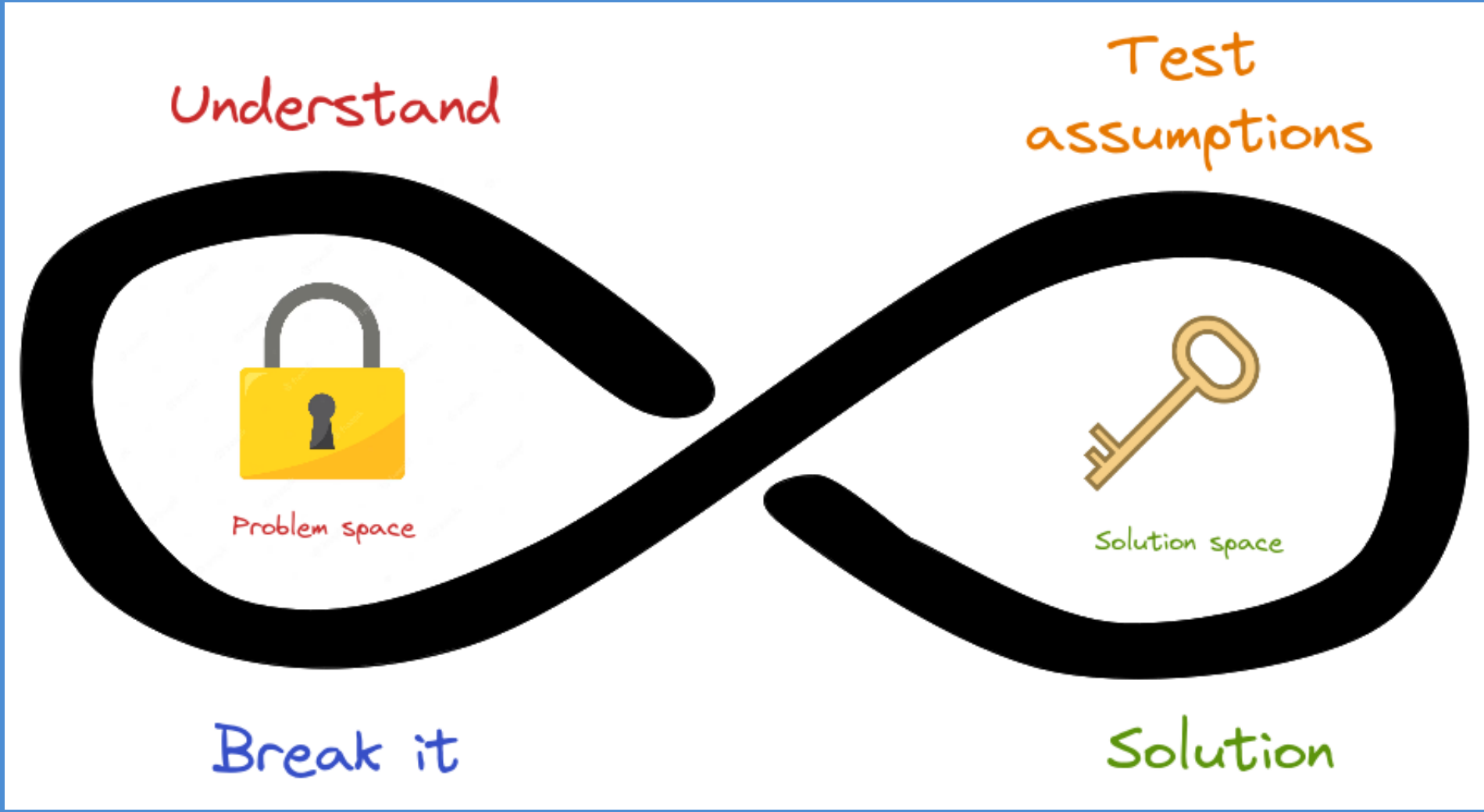
Solve Problems

With **First Principles Thinking**

1. Identify the **problem** you want to solve
2. Break it down into the fundamental **pieces**
3. Question and challenge your **assumptions**
4. Create a new **solution** from the ground up

Breaking down **complex** problems into **basic** elements and reassembling from the ground up can be valuable for moving forward.

Time for Action



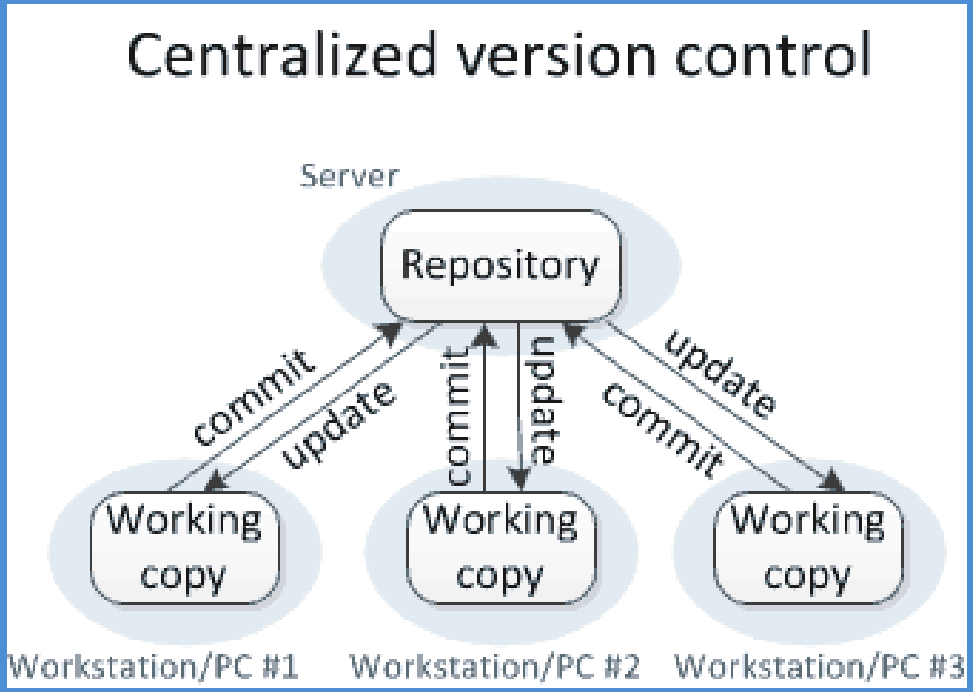
Problems in the Coding phase

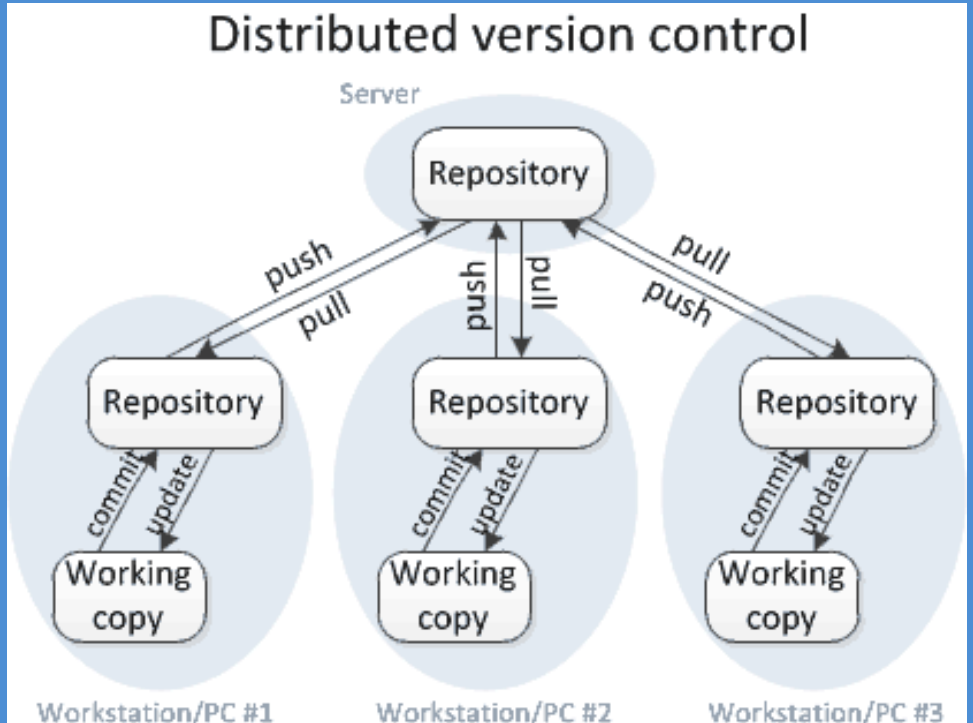
- How do you track your code changes ?
- What branching strategies should I follow ?
- How to enforce coding style/standards ?
- Is there any commercial/open source solution available ?

Solution

Tracking changes

- **Centralized** - each user gets their own working copy, but there is just one central repository. As soon as you commit, it is possible for your co-workers to update and to see your changes. (Ex: **Subversion**)
    - **You commit**
    - **They update**
- **Distributed** version control, each user gets their own repository and working copy. After you commit, others have no access to your changes until you push your changes to the central repository. When you update, you do not get others' changes unless you have first pulled those changes into your repository. (Ex: **Git**, **Mercurial**)
    - **You commit**
    - **You push**
    - **They pull**
    - **They update**





Version control

OSS

- Gitea
- Gitlab - Community edition + commercial
- Gogs

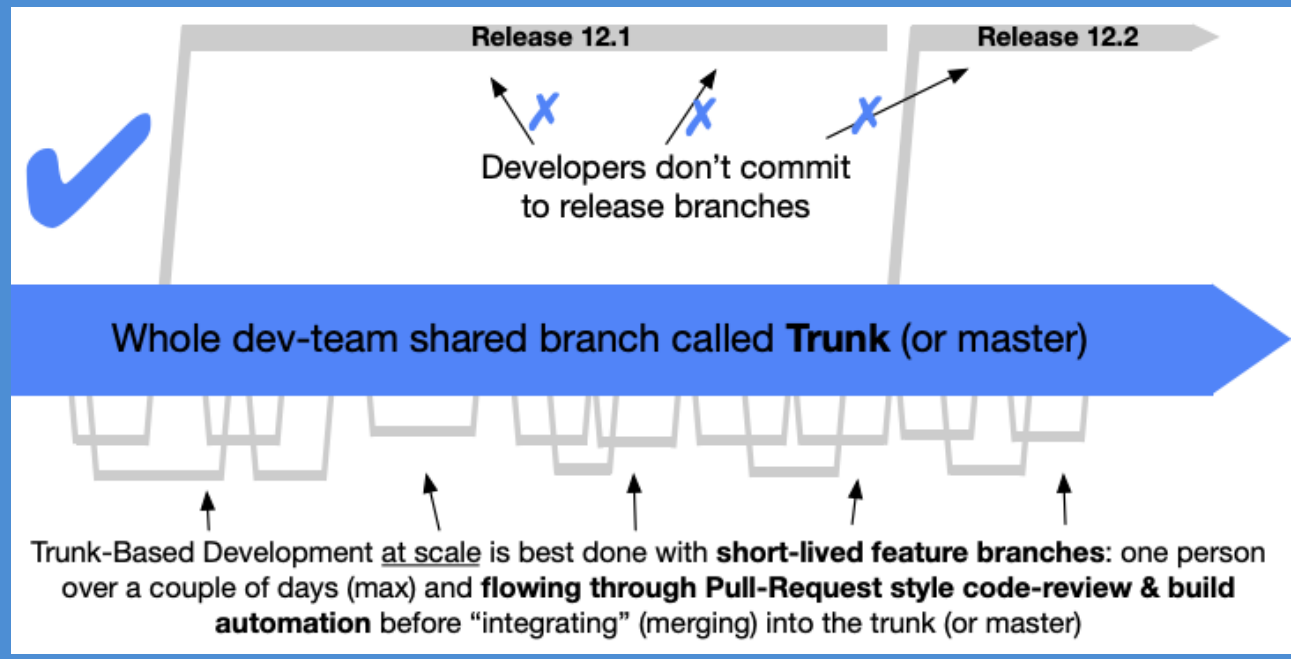
commercial

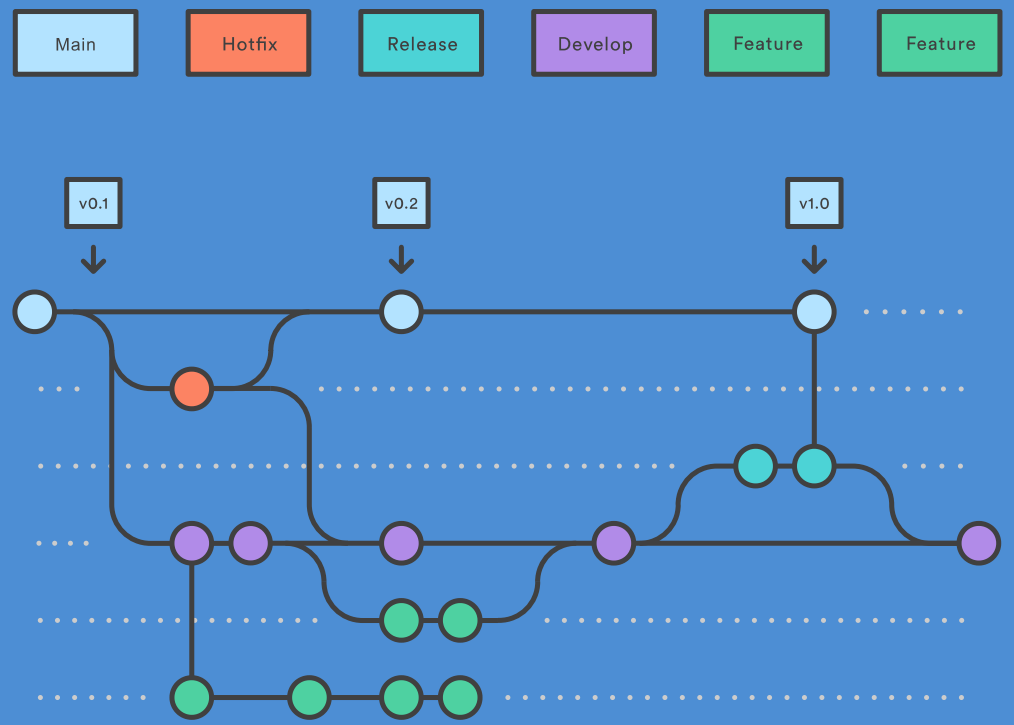
- Github
- Bitbucket
- Azure devops

Branching Strategies

- **Trunk based development**
- **Gitflow**

For more patterns for Managing source code branches refer here





Coding standards

- **Hooks** - *It fires custom scripts on a certain events during the execution of a git command, such as* `pre-commit` *and* `pre-push`.
  - Git hooks
  - pre-commit
  - husky
- **Linters** - is a tool used to flag programming errors, bugs, stylistic errors and suspicious constructs, for example in JS/TS suggesting to use `let/const` instead of `var`, collection of linters.
- **Formatter** - formats your code and enforces a consistent style across your code, collection of formatters.

Problems in the Build phase

- How do I compile my source code ?
- How to manage dependencies ?
- How to generate artifacts ?
- How do I version my artifacts so that I can rollback to ?
- Where do I store my artifacts ?

Solution

Build Automation tools

It is a process of automating an extensive range of tasks that one has to do in their day-to-day activity, right from source code to end-product.

- Downloading dependencies
- Compile source code into binary code
- Package code into an executable
- Generate documentation out of source code

- **Java** : Ant, Maven, Gradle
- **Javascript** :
  - Package manager - npm, Yarn, PNpm

    +
  - Build tools - Gulp, Webpack, Vite
- **Language agnostic/Multilingual** - Make, Bazel, Buck

Semantic Versioning (SemVer)

Semantic Versioning *or* ***SemVer*** *contain a set of rules and requirements that dictate how version numbers are assigned and incremented.*

Format : *MAJOR.MINOR.PATCH*

Repository manager

A repository manager is a dedicated server application designed to manage repositories of binary and artifact components such as *Maven*, *NPM*, *Container*, *RubyGems* ...

- Sonatype nexus - OSS + commercial
- JFrog - commercial
- Gitea
- Gitlab

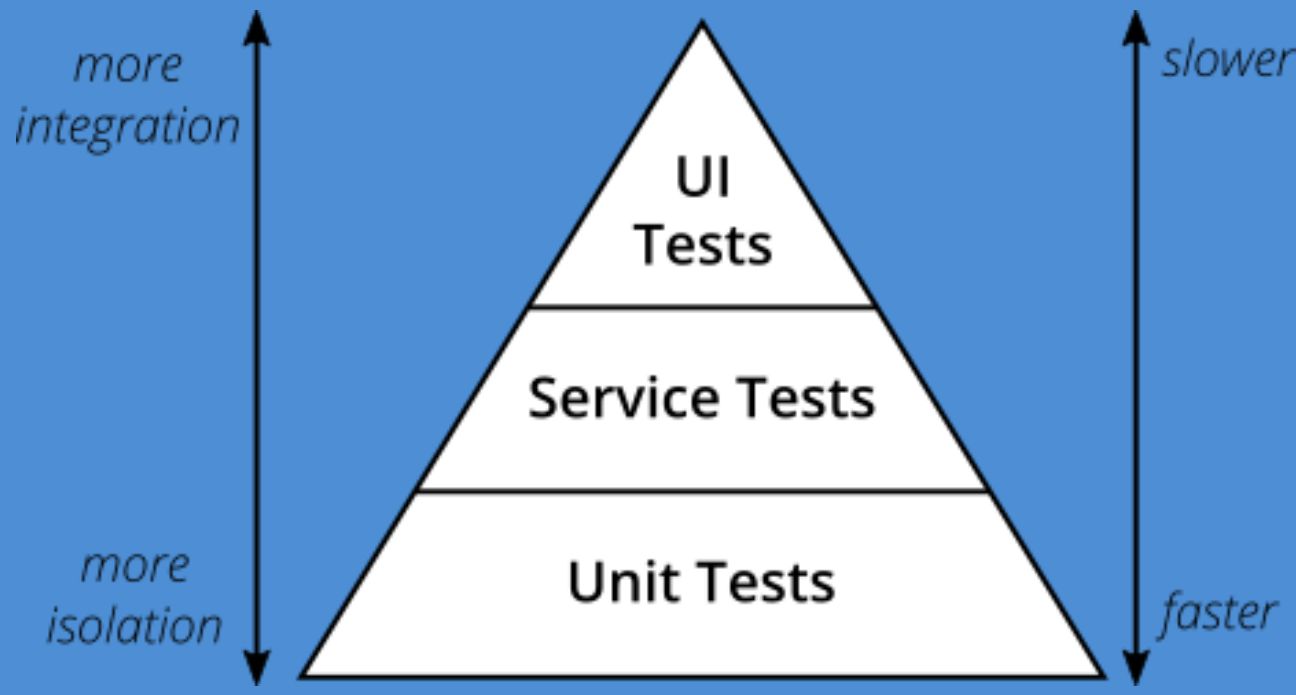
Problems in the Testing phase

- What should be tested at which level ?

Solution

Test pyramid

- Write tests with different granularity
- The more high-level you get the fewer tests you should have



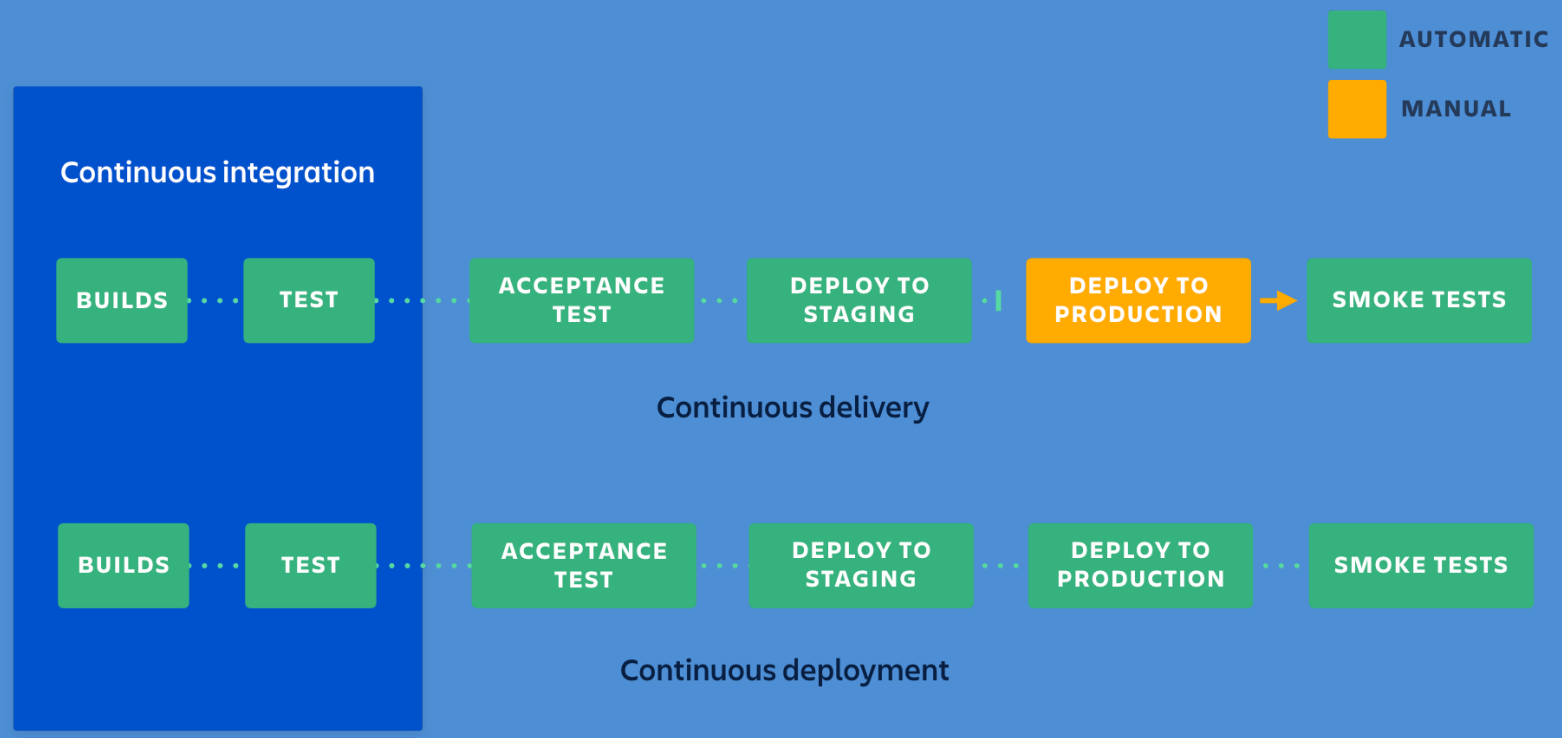
Problems in Release phase

- Why we need Continuous integration/delivery/deployment ?
- How to get the faster feedback cycle ?
- What are some of the oss/commercial service available ?
- How to ensure code quality ?
- Should I build artifacts for each env `QA`, `UAT`, `Prod` ? How should I promote it ?

Solution

Continuos integration/ delivery/ deployment

- Jenkins
- Gitlab
- Github actions
- Spinnaker CD
- Argo CD



Code quality

- Setting up Go/No-go quality gate
- Security rules
- Code coverage
- Code smells

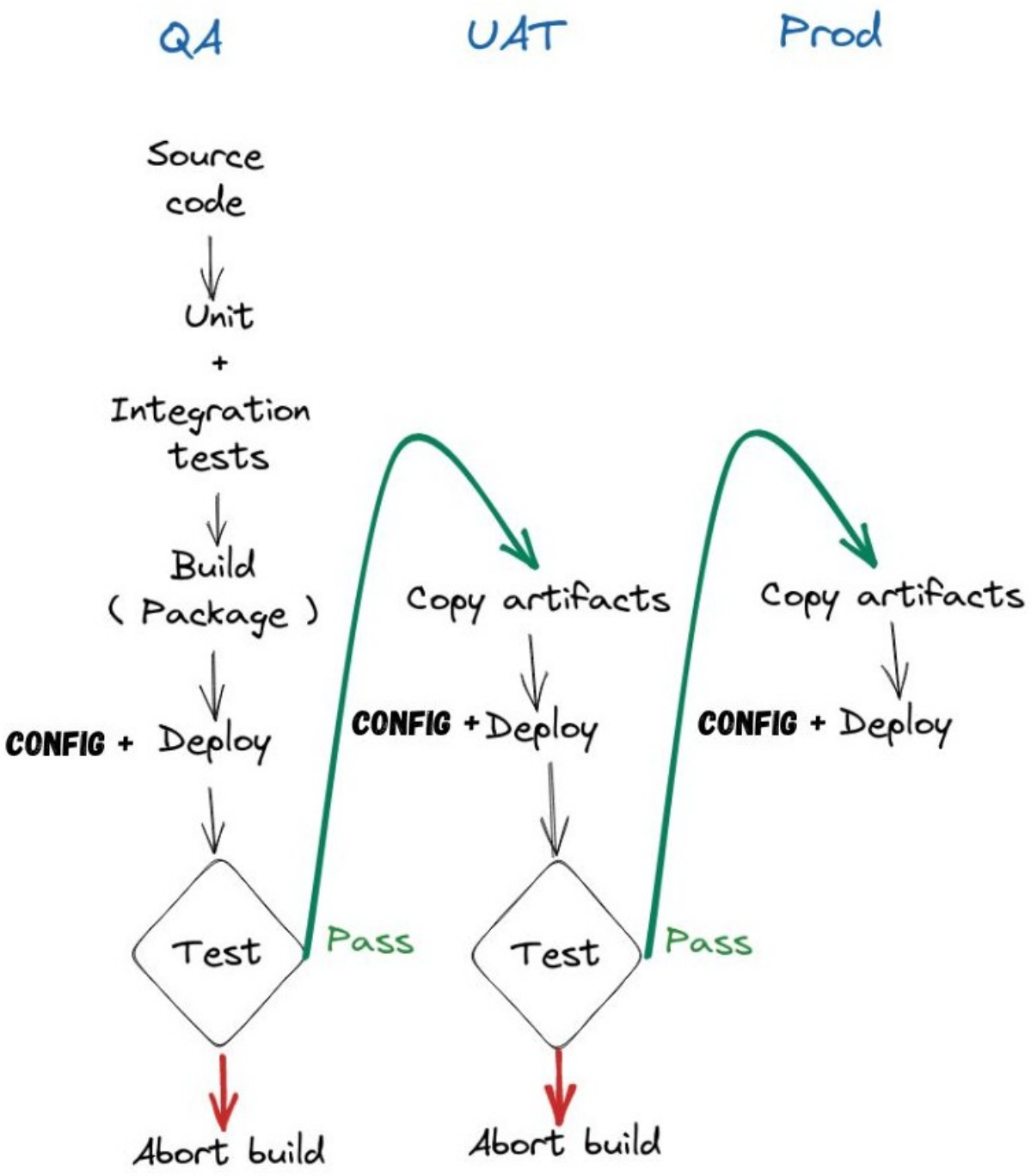
This process is called `Static Application Security Testing (SAST)`

Tools

- SonarQube - Community edition + commercial
- Deep Source - commercial

Build once and promote

***Only build packages once**. The binaries you release should be the same binaries that have been through the rest of your deployment pipeline, so you can be sure that what you release is what you tested.*



Problems in Deployment phase

- How to create and manage environment ?
- What is an Infrastructure as Code (IaC) ?
- How do I deploy an application to only a subset of users ?
- How do I rollback when there is an failure ?

Create and Manage environment

Infrastructure as code (IaC) tools allow you to manage infrastructure with configuration files rather than through a graphical user interface.

- ***Terraform*** - It is an ***Infrastructure provisioning*** tool, using it, we can build, destroy and manage infrastructure in a declarative way. And it takes a immutable approach and uses HCL.
- ***Pulumi*** *- is a modern IaC platform that allows you to use familiar programming languages and tools to build, deploy, and manage cloud infrastructure.*
- ***Ansible*** - It is a ***Configuration management*** tool, mainly used to configure systems, deploy software, and orchestrate more IT tasks using yaml.

Read more about the comparisons at Ansible vs. Terraform Demystified

Deployment strategies

Strategy	ZERO DOWNTIME	REAL TRAFFIC TESTING	TARGETED USERS	CLOUD COST	ROLLBACK DURATION	NEGATIVE IMPACT ON USER	COMPLEXITY OF SETUP
RECREATE version A is terminated then version B is rolled out	✗	✗	✗	■□□	■■■	■■■	□□□
RAMPED version B is slowly rolled out and replacing version A	✓	✗	✗	■□□	■■■	■□□	■□□
BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B	✓	✗	✗	■■■	□□□	■■□	■■□
CANARY version B is released to a subset of users, then proceed to a full rollout	✓	✓	✗	■□□	■□□	■□□	■■□
A/B TESTING version B is released to a subset of users under specific condition	✓	✓	✓	■□□	■□□	■□□	■■■
SHADOW version B receives real world traffic alongside version A and doesn't impact the response	✓	✓	✗	■■■	□□□	□□□	■■■

Problems in Monitoring phase

- What should be monitored ?
- How to monitor failures ?
- How do I visualize it and get notified ?

Solution

Collecting metrics and monitoring

Metrics *represent the raw measurements of resource usage or behavior that can be observed and collected throughout your systems*

- **Host-Based Metrics** - CPU, Memory, Disk space, and Processes
- **Application Metrics** - Error and success rates, Service failures and restarts, Performance and latency of responses, and Resource usage
- **Network and Connectivity Metrics** - Connectivity, Error rates and packet loss, Latency, and Bandwidth utilization

Monitoring *is the process of collecting, aggregating, and analyzing those values to improve awareness of your components' characteristics and behavior*

Tools

- Prometheus
- InfluxDB
- Datadog
- New relic
- Grafana - Visualization

Log monitoring

Main purpose is to observe the streams of logs generated by the applications to provide information and trigger alerts if something affects system performance and health.

- Loki
- Elastic search- ELK
- Sentry

Alerts

Automated alerts are essential to monitoring. It is process of sending notification message informing about a change of state, typically signifying a potential problem in the form of email, SMS, Slack notification, or a ticket.

References

- DevOps Vs. SRE: Competing Standards or Friends? (Cloud Next '19)
- How Netflix Thinks of DevOps
- DevOps Culture at Amazon
- Enterprise DevOps: Why You Should Run What You Build
- Full Cycle Developers at Netflix — Operate What You Build
- Atlassian Devops Guide
- First Principles: The Building Blocks of True Knowledge
- First Principles for Software Engineers
- Version control concepts
- Atlassian Gitflow workflow
- Trunk based development
- Practical test pyramid
- Build promotion
- Deployment strategies
- An Introduction to Metrics, Monitoring, and Alerting
- Effective Monitoring and Alerting

Cloud native

containerization

THREAT ANALYSIS

ORCHESTRATION

OBSERVABILITY

microservices

CYBER SECURITY

CHAOS ENGINEERING

Thank you

gitops

12 FACTOR AUTHENTICATION

SERVERLESS

ZERO TRUST

service mesh