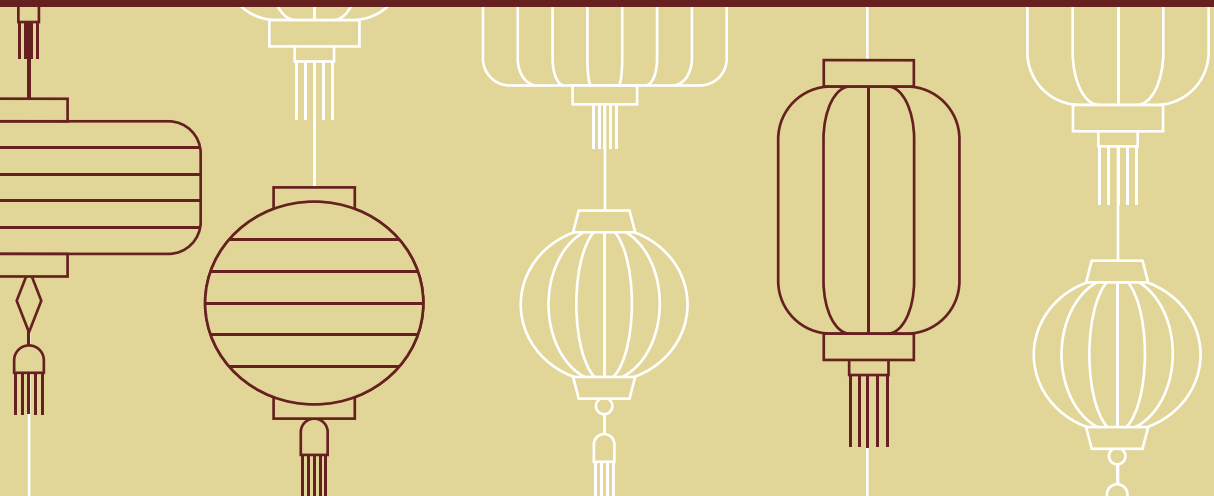
Introduction to DevOps

MADE BY:

ARYAN KHURANA



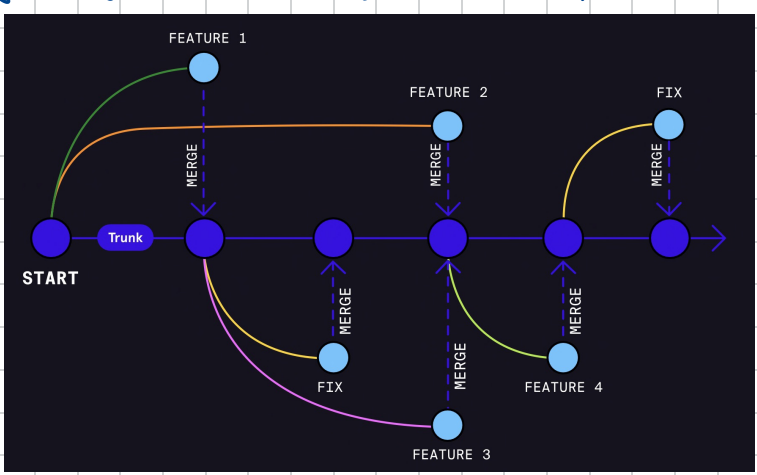
Deployment is the general process of making a piece of software available to its users.

INFRASTRUCTURE MANAGEMENT:

- Software is stored and executed on servers.
  - ↳ Computers that respond to requests.
  - ↳ Requests are from other computers (client).
  - ↳ Usually over the internet.
- Operations (ops) team manages a company's infrastructure.

VERSION CONTROL SYSTEMS:

- Tools designed to manage different versions of a file or project.
- Branching is the process of creating a copy of the source code.
- Merging is the process of combining the changes in one branch with another.
- Version control systems can be synchronized with project management tools.



TESTING.

- This is an essential component. It ensures new features integrate with the existing ones.
- Types:
  1) Unit Test: Evaluates smallest unit of code (eg. Function).
  2) Integration Test: Checks how well the units work with one another.
  3) Acceptance Test: UX alignment with business requirement evaluation
  4) End-to-End Test: App behaviour testing.

ENVIRONMENTS:

1) Local Development Environment: Where software is first written and tested, typically on a developer's own computer.
2) Integration Environment: Where software changes are merged using a version control system.
3) Quality Assurance: Where tests are executed to ensure the functionality and usability of each feature.
4) Staging Environment: Software is performance tested in a production like environment.
5) Production Environment: Software is available to real users.

THE DEV-OPS CULTURE:

THE TRADITION:

1) The development team writes an app's features.
2) The ops team creates and maintains the infra that the app runs on.
3) They own different environments. This can lead to bugs that are difficult to resolve.
4) Handoffs between teams take time.
5) Information is siloed.

CURRENT CULTURE:

1) DEV + OPS
2) Systems level Thinking
3) Continuous Experimentation and learning
4) Feedback loops.
5) Shared-responsibility model

INFRASTRUCTURE MANAGEMENT

- Scalability is a system's ability to add resources to keep up with growing demand.

1) VERTICAL Scaling: Adding computing resources such as increasing network speeds, storage or RAM.
2) HORIZONTAL Scaling: Adding more servers/nodes that each run the app. Load balancer is used to distribute the workload.

- Elasticity is the ability to automatically add or subtract resources to accomodate fluctuating demand.

VIRTUALIZATION:

Virtualization technology allows many virtual machines (VM's) to run on one physical computer.
Each VM has a distinct environment, own OS, dependencies and users.
- ↳ Efficient Horizontal Scaling
- ↳ Each server can be utilized closer to its full capacity by being split into several VM's.

CONTAINERIZATION:

Containerization is a form of virtualization in which users create virtual environments called containers.
- ↳ Include app instances and dependencies.
- ↳ Do not include OS.
- ↳ Smaller and faster than VMs.
- ↳ less resources than VMs.

ORCHESTRATION:

Automated configuration, management and coordination of infrastructure.
Ex.) Docker → Ability to create and control individual containers.
Kubernetes → Controls many containers.
- ↳ Deploy containers on many servers. → Restart failed containers
- ↳ Update without downtime → Horizontal scaling of containerized apps

INFRASTRUCTURE AS CODE:

IaC is the act of defining infrastructure properties in configuration files. Traditionally, configuration drifts would occur due to human errors.

Since config. files are machine readable, it enables for easier automation.

- Immutable Infrastructure : Servers are not changed once they are created. Instead, new servers are created and old ones are destroyed.

- This is only possible because of VMs and containers.
- Lower business costs, better UX.

MONITORING

Monitoring helps teams understand the state of their systems based on gathered data.

METRICS: Metrics express a value relevant to a system at a specific point in time.

1) **Latency:** Time between the start of an event to its completion. This is a key indicator of performance.
2) **Traffic:** Amount of system usage over time.
3) **Errors:** An error is an invalid state that our system has reached.
4) **Saturation:** Load on our system's resources.

SERVICE LEVEL OBJECTIVE (SLO): SLO is a valid range of measurements for a metric. This is the goal that we wanna achieve.

SERVICE LEVEL INDICATOR (SLI): SLI is the current measurements of a metric related to an SLO. This is where we are right now.

SERVICE LEVEL AGREEMENT (SLA): This is a contract with consumers. SLA binds the business to the level of expected service promised to a customer. Breaking an SLA can have legal consequences.

ALERTING: An alert is a notification informing about a change of state, usually a problem.

- Tickets
- Email Alerts
- Slack Alerts
- SMS / phone calls

* Only alert when immediate human intervention is required.
* Alert based on customer facing issues.
* Set clear ways to indicate urgency.
* Ensure an alert is not a copy of another.

OBSERVABILITY: Observability is the degree to which a system's information can be used to locate and fix a problem.

RESILIENCY

- Resiliency means fast recovery from a degraded system state.

→ **Internal Problems:** Problems that arise from within the components of a system that we control.

→ **External Problems:** Problems that we have from dependencies from other parties.

→ **Malicious Actors:** Problems that stem from other people that seek to disrupt or exploit our services.

- **RECOVERY TIME OBJECTIVE (RTO):** RTO is the amount of time an application can be available before it causes significant harm to the business.

- **RECOVERY POINT OBJECTIVE (RPO):** RPO is the acceptable amount of data loss after a system outage.

→ **TESTING RESILIENCY:**

1) Penetration Testing → Involves simulating cyber attacks to try to exploit security vulnerabilities. Allows us to fix holes in our security.

2) Load Testing → Seeks to replicate situations in which the system is under heavy use.

→ **CHAOS ENGINEERING:** Practice of performing experiments on a system in order to test its resiliency. Chaos engineering emerged at Netflix.

CI/CD PIPELINES

- **Bottlenecks of manual deployment:**
  1) Version control management
  2) Testing
  3) Infrastructure and environments

- **Continuous Testing:**

This involves automatically triggering tests to be executed once an application is built in a new environment.

↳ Unit + Integration Testing ⇒ During Development

↳ Acceptance + End-to-End Testing ⇒ Before production

- **Continuous Integration:**

- Merging source code changes on a frequent basis.
- Building and Testing the changes in an automated process.

↳ Main focus is frequency. Instead of building a feature on one branch, the changes are more frequent which leads to less merge conflicts.

- **Continuous Delivery:**

It is the automated process of preparing new versions of an app to be deployed into the production environment.

Automates → Environment configuration through containerization and Infra as code.
→ Deployment to intermediary environments.
→ Further testing.

- **Continuous Deployment:**

Automatic process of deploying a project to the production server after it has been tested in testing and staging environments.

→ If previous stages were successful, the deployment is approved automatically.
→ Monitoring tools identify bugs in production.