What is DevOps according to you, give a sample from your past experience?

The term “DevOps” is a combination of the words “development” and “operations.” DevOps helps speed up the delivery of applications and services. It allows organizations to better serve their customers and be more competitive in the market. In simple terms, DevOps aligns between development and IT performance through better communication and collaboration.

DevOps takes on a culture where collaboration between development, operational, and business groups is considered a critical aspect of travel. It’s not just about tools, as DevOps in the organization creates a continuous number of customers. Tools are one of its pillars, along with people and processes. DevOps increases organizational capacity to deliver high-quality solutions at a faster rate. Automatically execute all processes, from construction to delivery, system or product.

**Shared Load Culture**

DevOps incorporates a culture created by adopting Lean-Agile values, policies and practices across the framework:

* **Collaboration and Integration** – DevOps relies on the ability of Agile teams and IT Operations to work together effectively in a sustainable manner, ensuring that solutions are built and delivered quickly and reliably. This is used, in part, to incorporate staff and skills across ART.
* **Accident Tolerance** – DevOps requires tolerance for failure and recovery quickly, and it is worth taking the risk.
* **Automated Infrastructure**– Automated infrastructure enables development and functionality to operate independently.
* **Sharing Information** – Sharing experiences, practices, tools, and learning across groups, ART and the wider organization are encouraged.
* **Automation in Mind**– DevOps relies heavily on automation to provide speed, consistency, and duplicate processes and environment architecture.

<https://engineerbabu.com/blog/devops-digital-transformation-guide-for-non-tech-founder/>

Explain the workflow from “code-to-production” from your past engagements and what improvements that you want to make?

Clone a copy of the source code and create a new branch

Run the code locally

Make code changes

Submit changes for review

Run automated tests

Merge and release

Deploy

A key point of frustration for many developers is a slow and inflexible release cycle. The slower the release cycle, harder it is for a developer to move features from inception to production. This creates an ever increasing backlog that a developer needs to manage and sustain, when they could be building new features. LinkedIn's solution is to this challenge is our shift to an increasingly agile release cycle. This allows our developers to build the various part of a feature in an iterative manner and can help make the development cycle easy to manage.

Problem Space

When a developer aspires to push a feature to production servers, rigorous certification testing is typically required. This release ritual often changes each time; usually comprising of running tests, manually analyzing failures, re-testing the failures and iterating until all the bugs in the release build are fixed. A lower release frequency increases the susceptibility to issues in production as the changeset is proportionately larger.

in order to achieve fully automated, high quality, and low latency releases, we identified the following as key parts of our new automation strategy:

Test services in isolation

Execute tests at earlier part of development cycle

Team ownership of quality

<https://engineering.linkedin.com/developer-happiness/getting-code-production-less-friction-and-high-quality>

What’s an overlay network?

An “overlay network” is a virtual network that runs on top of a different network. Devices in that network are unaware that they are in an overlay. Traditional VPNs, for instance are overlay networks running over Internet.

The term “overlay” has come to be used extensively (instead of VPN) only after technologies different than PPTP or L2TP have been developed to run virtual networks in Cloud environments. For those environments, protocols like VXLAN or GENEVE have been developed to address specific needs.

The overlay network driver creates a distributed network among multiple Docker daemon hosts. This network sits on top of (overlays) the host-specific networks, allowing containers connected to it (including swarm service containers) to communicate securely when encryption is enabled. Docker transparently handles routing of each packet to and from the correct Docker daemon host and the correct destination container.

which CNI plugin that you have used in the past for the kubernetes cluster?

Amazon EKS supports native VPC networking with the Amazon VPC Container Network Interface (CNI) plugin for Kubernetes. Using this plugin allows Kubernetes pods to have the same IP address inside the pod as they do on the VPC network

<https://santanderglobaltech.com/en/using-docker-overlay-networks-configuration-guide/>

<https://docs.aws.amazon.com/eks/latest/userguide/pod-networking.html>

sample kubernetes manifest with 2 RBAC roles, first is Cluster ReadOnly Access and 2 is only Pod Readonly access

Cluster readonly access

POD Readonly :

kind: Role

apiVersion: rbac.authorization.k8s.io/v1

metadata:

namespace: default

name: pod-reader

rules:

- apiGroups: [""] # "" indicates the core API group

resources: ["pods"]

verbs: ["get", "watch", "list"]

# This role binding allows "jane" to read pods in the "default" namespace.

kind: RoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: read-pods

namespace: default

subjects:

- kind: User

name: jane

apiGroup: rbac.authorization.k8s.io

roleRef:

kind: Role

name: pod-reader

apiGroup: rbac.authorization.k8s.io

<https://docs.bitnami.com/tutorials/configure-rbac-in-your-kubernetes-cluster/>

<https://sysdig.com/blog/kubernetes-security-rbac-tls/>

Write a simple dockerfile that builds a Spring boot service. (15 Marks) (Medium) (30 Min)

<https://stackabuse.com/dockerizing-a-spring-boot-application/>

Write a kubernetes manifest to allow the external traffic via a couple of approaches, one is via load balancer and another via nodeport.

Load Balancer

**apiVersion**: v1

**kind**: Service

**metadata**:

**name**: my-service

**spec**:

**selector**:

**app**: MyApp

**ports**:

- **protocol**: TCP

**port**: 80

**targetPort**: 9376

**clusterIP**: 10.0.171.239

**type**: LoadBalancer

**status**:

**loadBalancer**:

**ingress**:

- **ip**: 192.0.2.127

Nodeport

**apiVersion**: v1

**kind**: Service

**metadata**:

**name**: my-service

**spec**:

**type**: NodePort

**selector**:

**app**: MyApp

**ports**:

*# By default and for convenience, the `targetPort` is set to the same value as the `port` field.*

- **port**: 80

**targetPort**: 80

*# Optional field*

*# By default and for convenience, the Kubernetes control plane will allocate a port from a range (default: 30000-32767)*

**nodePort**: 30007

<https://kubernetes.io/docs/concepts/services-networking/service/#loadbalancer>