**Sample Projects:**

**1: React App Deployment with Docker**

**Description**: Dockerize a React application using a multi-stage Docker build, and automate the process of building and deploying the Docker image to Docker Hub using bash scripts in jenkins pipeline.

**2: Web Application Deployment with AWS EC2, Load Balancer, and Route 53**

**Description:**

Deploy a web application using AWS EC2 instances, Application Load Balancer, and Route 53. Use EC2 instances to host the applications, while the Application Load Balancer distributes traffic across multiple instances for increased availability and scalability. Create Route 53 DNS service to route traffic to the appropriate EC2 instances.

**3: Continuous Integration for a Node.js Application using Git, Jenkins, and**

**AWS Elastic Beanstalk**

**Description:**

Set up a Continuous integration pipeline using Jenkins to build and test the Node.js application in Git. Use AWS Elastic Beanstalk service to automate the deployment and scaling of web applications.

**4: Infrastructure as Code using Terraform and AWS VPC**

**Description:**

Create a Terraform template to create AWS VPC by incorporating attributes like CIDR, private and public subnets,route table Internet gateway etc.

**5: Container Orchestration with Kubernetes on AWS EKS**

**Description:**

Configure a Kubernetes cluster on AWS Elastic Kubernetes Service (EKS) to orchestrate ,deploy and scale containerized applications on it.

**6.Continuous Integration and Deployment of Node.js Application with Jenkins, AWS EC2 and Docker.**

**Description:** Set up a Continuous integration and deployment pipeline using Jenkins to dockerize and test the Node.js application and push the docker image in docker hub repo.

**7.Infrastructure Monitoring and Alerting with AWS CloudWatch.**

**Description:** Configure CloudWatch alarms to trigger notifications (e.g. email, SMS) when the thresholds are breached

**8.Infrastructure Automation with AWS Lambda**

**Description:**1.Create a sample AWS S3 bucket and define an event trigger that calls a Lambda function.

2.Create a Lambda function that automates a predefined task (e.g. resizing images, encrypting files, copying files to another location)

3.Test the automation by uploading files to the S3 bucket and verifying that the Lambda function is triggered and automates the predefined task.

**9. Infrastructure as Code with AWS CloudFormation**

**Description:** Deploy the CloudFormation stack to create the infrastructure(ec2,security groups,iam roles)

**10. Continuous Integration and Deployment with Docker, Jenkins, AWS EKS, and establish monitoring with Prometheus, and Grafana**

**11. Create dockerfile and docker compose file for the java, python, nodejs applications**

**12. Create an Auto Scaling group** using the AWS Management Console and configure it to launch EC2 instances in response to changes in demand and Use Amazon CloudWatch to monitor the performance of the Auto Scaling group and the EC2 instances and troubleshoot any issues that arise.

**13. How to deploy your React app in s3**

**14. AWS S3 Event Triggering Shell Script Used by Netflix, Airbnb, Adobe, Expedia, and Others**

**15. Create a jenkins freestyle project for a nodejs application by building and deploying it to ec2 instances .Create dockerfile and docker compose file for build and deployment.**

**16. Create a 3 tier project in AWS - Presentation tier, Application tier and database tier (setup VPC, subnet, IG, NAT gateway, route table, ec2 launch template, auto scaling group)**

**17. Jenkins Multi Branch CICD Pipeline using dev and prod environment**

**18.Create a continuous integration and deployment for Dockerised Node app to AWS Elastic Beanstalk with AWS CodePipeline**

**19. Write a shell script to report the usage of AWS resources in your project**

**20. Create a Nginx AMI in aws**

**Shell scripts & Python scripts**

1.Write a program for displaying 1 to 10 numbers using For loops

2.Write a program using If else in Shell Scripting by comparing 3 numbers

3.Rename files in a directory

4.Prompt user for input and perform calculation - add, sub, multiply, divide

5.Display system information

6.Count the number of lines in a file

7.Copy files from one directory to another

8.Check if a website is up or down

9.Convert all files in a directory to lowercase

10.System monitoring scripts, log analysis scripts and Log rotation scripts

### **Best Practices**

### **1. Infrastructure as Code (IaC):**

* Step 1: Clearly understand the requirements of the infrastructure needed for the application.
* Step 2: Choose the appropriate IaC tool (Terraform or AWS CloudFormation) based on the organization's preferences.
* Step 3: Write modular and reusable code, starting with essential components like VPC, subnets, and security groups.
* Step 4: Test the IaC scripts locally using tools like terraform validate or AWS CloudFormation's aws cloudformation validate-template.
* Step 5: Deploy the infrastructure to a sandbox environment and validate its correctness.
* Step 6: Explain how you handle secrets and sensitive information securely in your IaC scripts.

### **2. Continuous Integration/Continuous Deployment (CI/CD):**

* Step 1: Define a simple CI/CD pipeline with source, build, test, and deploy stages using AWS CodePipeline.
* Step 2: Integrate AWS CodeBuild to build artifacts and run tests.
* Step 3: Configure AWS CodeDeploy for deploying the application.
* Step 4: Test the CI/CD pipeline with a sample application and monitor each stage.
* Step 5: Explain how you handle rollbacks and versioning in the CI/CD process.
* Step 6: Showcase how you integrate automated testing into the pipeline.

### **3. Monitoring and Logging:**

* Step 1: Define key performance metrics relevant to the application and set up CloudWatch Alarms.
* Step 2: Configure centralized logging using CloudWatch Logs or another logging solution.
* Step 3: Create a CloudWatch dashboard to visualize important metrics.
* Step 4: Discuss how you would handle monitoring for auto-scaled instances.
* Step 5: Explain how you set up notifications for critical events.

### **4. Security and Compliance:**

* Step 1: Define IAM roles and policies based on the principle of least privilege.
* Step 2: Implement encryption using AWS KMS for data at rest and in transit.
* Step 3: Demonstrate understanding of AWS Organizations for managing multiple AWS accounts securely.

### **5. Containerization and Orchestration:**

* Step 1: Choose between ECS or EKS based on the application requirements.
* Step 2: Create a Dockerfile for a sample application and push it to Amazon ECR.
* Step 3: Deploy the containerized application using ECS or EKS.
* Step 4: Configure auto-scaling for the containers.

### **6. Networking:**

* Step 1: Design and configure a VPC with appropriate subnets, route tables, and security groups.
* Step 2: Set up a VPN or Direct Connect for secure on-premises connectivity.
* Step 3: Configure Route 53 for DNS management, including setting up a failover setup.
* Step 4: Explain how you would optimize networking for high availability and low latency.
* Step 5: Discuss strategies for securely connecting different VPCs.

### **7. Automation and Scripting:**

* Step 1: Showcase proficiency in writing scripts for automation.
* Step 2: Develop scripts for routine operational tasks or resource management.
* Step 4: Explain how you handle error handling and logging in your scripts.
* Step 5: Showcase any experience with infrastructure as code tools for automation beyond basic scripting.

### **8. Cost Optimization:**

* Step 1: Implement AWS Budgets to set cost limits and alerts.
* Step 2: Use AWS Cost Explorer to analyze and understand cost breakdowns.
* Step 3: Identify areas for cost optimization, such as reserved instances or spot instances.
* Step 4: Discuss strategies for controlling costs during development and testing phases.
* Step 5: Explain how you monitor and manage costs over time.

### **9. Collaboration and Documentation:**

* Step 1: Use AWS Systems Manager to automate routine operational tasks across instances.
* Step 2: Demonstrate effective documentation practices using tools like AWS SSM Documents or Confluence.
* Step 3: Explain how you ensure documentation stays up-to-date with changes in the infrastructure.
* Step 4: Discuss strategies for collaborative work within a DevOps team.
* Step 5: Showcase any experience with version control systems for infrastructure code.

### **General Tips:**

* Communication: Clearly articulate your thought process and decisions.
* Problem-Solving: Approach each task systematically, breaking it down into manageable steps.
* Adaptability: Be ready to adapt your solutions based on feedback or changing requirements.
* Best Practices: Emphasize adherence to best practices for security, scalability, and reliability.
* Demonstration: Whenever possible, showcase your work through practical demonstrations.

## **How to evaluate your DevOps skills, how much you need to know to pass interview to DevOps engineering?**

## **The basics about DevOps and responsibilities:**

DevOps is a set of practices that help automate and integrate the processes between development teams and teams responsible for infrastructure (operations) so that they can build, test, and release faster and more reliably. (Please see details <https://www.atlassian.com/devops>)

The main goal of the DevOps approach is to remove the “wall” between the development team and the operations team (operations are also called: System Administration, System Engineering, Infrastructure, IT) and to increase the speed of releases. The “wall” is formed because Dev and Ops have different goals. Developers aim to release as often as possible, and Operations aims to reduce the number of infrastructure fails or keep the environment stable and secure. The DevOps approach brings teams, goals, and risks together.

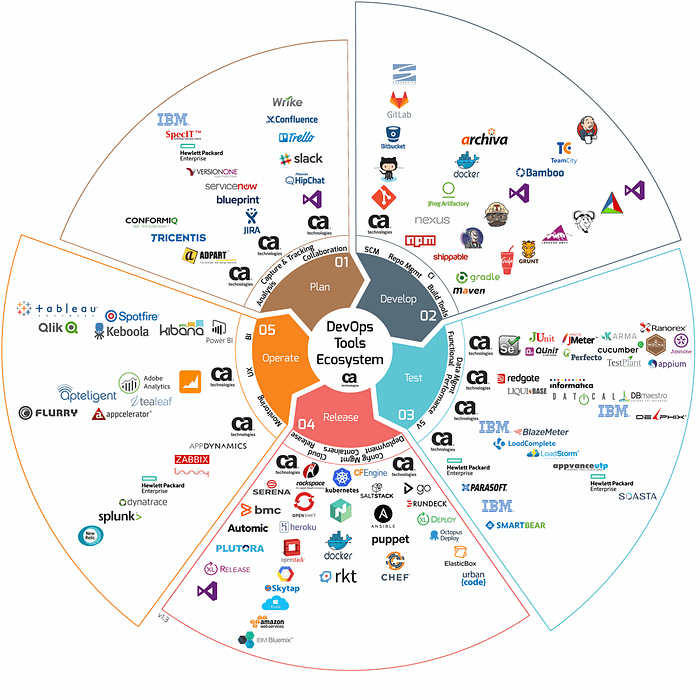
The main DevOps practices are:

* Infrastructure as Code
* Continuous Integration
* Continuous Delivery
* Continuous deployment
* Continuous Testing
* Continuous Monitoring

These practices are also the primary responsibilities of a DevOps engineer. Some practices can be divided between engineers. For example, QA may be responsible for Continuous Testing and Security Engineer for Continuous Monitoring. As a rule, the engineer who got the DevOps role (it’s not customary to say “DevOps engineer”, but the market has already taken over, sounds as vague as “Agile engineer”) is responsible for everything that happens from the moment the development team pushes the code to the repository or launches a feature. Which means that DevOps can be responsible for CI/CD pipelines, project environment, basic monitoring etc.

## **How many tools are there in a DevOps domain? What exactly do you need to know?**

PROD grade infrastructure, IaC, CI/CD pipelines, monitoring, security, etc… Are an entire ecosystem of tools. Now there are more than 100 popular names. Some of them you need to know deeply and be able to configure. Some just need to be configured once by reading the short documentation.



**what to choose, how many tools do you need to know, and what should be demonstrated at in an interview?**

Engineers who migrate to DevOps, for example from system administration, learning technologies, fear that they will be tied to one thing and for another project their experience will not be relevant. For example: you learned **Ansible** but for the project **Terraform** is required.



**How many tools/platforms and how deep do you need to know?**

To make selection easier, all DevOps tools can be divided into domains:



DevOps tools by domains (I made this difficult infographic for two days)

The picture shows 30 most popular tools divided by their respective domains, you can add domains or tools as you need. Trying to learn everything is pointless, long and demotivating. **It’s better to choose one or two tools from each domain and compile a small, personal competency matrix, from tools you really want to know, for instance**:

Now we have the following matrix:

* **IaC**: Terraform — **Advanced**
* **CM**: Ansible — Intermediate
* **Cloud**: AWS — Intermediate
* **CI/CD**: Jenkins — Expert
* **Scripting**: Python, Bash — Novice
* **Containerisation**: Kubernetes — Intermediate
* **Monitoring**: ELK, Prometheus — Novice
* **OS**: Linux — **Advanced**
* **SQL**: MySql - optional