DevOps & Cloud Essentials Cheatsheet

A concise reference guide for essential DevOps and Cloud concepts, commands, and best practices, covering various tools and platforms.

Core DevOps Concepts

DevOps Principles

Culture: Foster collaboration and communication between development and operations teams.

Automation: Automate repetitive tasks and processes to improve efficiency and reduce errors.

Measurement: Track key metrics to monitor performance and identify areas for improvement.

Sharing: Share knowledge and best practices across teams to promote continuous learning.

Continuous Improvement: Continuously seek ways to optimize processes and improve overall system performance.

Key Practices

Continuous Integration (CI)	Automate the integration of code changes from multiple developers into a shared repository.
Continuous Delivery (CD)	Automate the release process, ensuring that code changes are reliably and frequently deployed to production.
Infrastructure as Code (IaC)	Manage and provision infrastructure through code, enabling automation, version control, and repeatability.
Monitoring and Logging	Implement robust monitoring and logging systems to track system health, performance, and identify potential issues.

DevOps Lifecycle Stages

Plan -> Code -> Build -> Test -> Release -> Deploy -> Operate -> Monitor -> Plan (cycle repeats)

Each stage involves specific tools, practices, and collaboration between teams.

Cloud Computing Essentials

Cloud Service Models

Infrastructure as a Service (IaaS): Provides access to computing resources (virtual machines, storage, networks).

Platform as a Service (PaaS): Offers a platform for developing, running, and managing applications without managing the underlying infrastructure.

Software as a Service (SaaS): Delivers software applications over the internet, on demand.

Cloud Deployment Models

Public Cloud	Cloud infrastructure owned and operated by a third-party provider.
Private Cloud	Cloud infrastructure used exclusively by a single organization.
Hybrid Cloud	Combination of public and private clouds, allowing data and applications to be shared between them.
Community Cloud	Cloud infrastructure shared by several organizations with similar interests.

Key Cloud Concepts

Scalability: Ability to increase or decrease resources as needed to handle changing workloads.

Elasticity: Ability to automatically provision and deprovision resources based on real-time demand.

Resilience: Ability to withstand failures and maintain availability through redundancy and fault tolerance.

Pay-as-you-go: Pricing model where you only pay for the resources you consume.

Containerization with Docker

Basic Docker Commands

`docker run [image]`	Create and start a container from an image.
`docker ps`	List running containers.
`docker stop [container_id]`	Stop a running container.
`docker images`	List available images.
`docker build -t [image_name] .`	Build an image from a Dockerfile in the current directory.
`docker pull [image]`	Download an image from a registry (e.g., Docker Hub).

Dockerfile Instructions

`FROM [image]` - Specifies the base image for the container.

`RUN [command]` - Executes a command during the image build process.

`COPY [source] [destination]` - Copies files from the host to the container.

`EXPOSE [port]` - Exposes a port from the container.

`CMD [command]` - Specifies the command to run when the container starts.

`WORKDIR [path]` - Sets the working directory inside the container.

Docker Networking

`docker network create [network_name]` - Create a new network.

`docker network connect [network_name] [container_id]` - Connect a container to a network.

`docker port [container_id]` - List port mappings for a container.

Orchestration with Kubernetes

Kubernetes Concepts

Pod: The smallest deployable unit in Kubernetes, representing a single instance of an application.

Deployment: Manages the desired state of pods, ensuring the specified number of replicas are running.

Service: Exposes an application running in a set of pods as a network service.

Namespace: Provides a way to logically isolate resources within a cluster.

Node: A worker machine in Kubernetes, either a virtual or physical machine.

kubectl Commands

`kubectl get pods`	List all pods in the current namespace.
`kubectl create deployment [name] --image=[image]`	Create a new deployment.
`kubectl expose deployment [name] --port=[port] --target-port=[target_port]`	Expose a deployment as a service.
`kubectl scale deployment [name] --replicas=[count]`	Scale a deployment to the specified number of replicas.
`kubectl apply -f [filename.yaml]`	Apply a configuration file to create or update resources.

Basic YAML Structure

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-app
        image: my-app-image
        ports:
        - containerPort: 8080
```