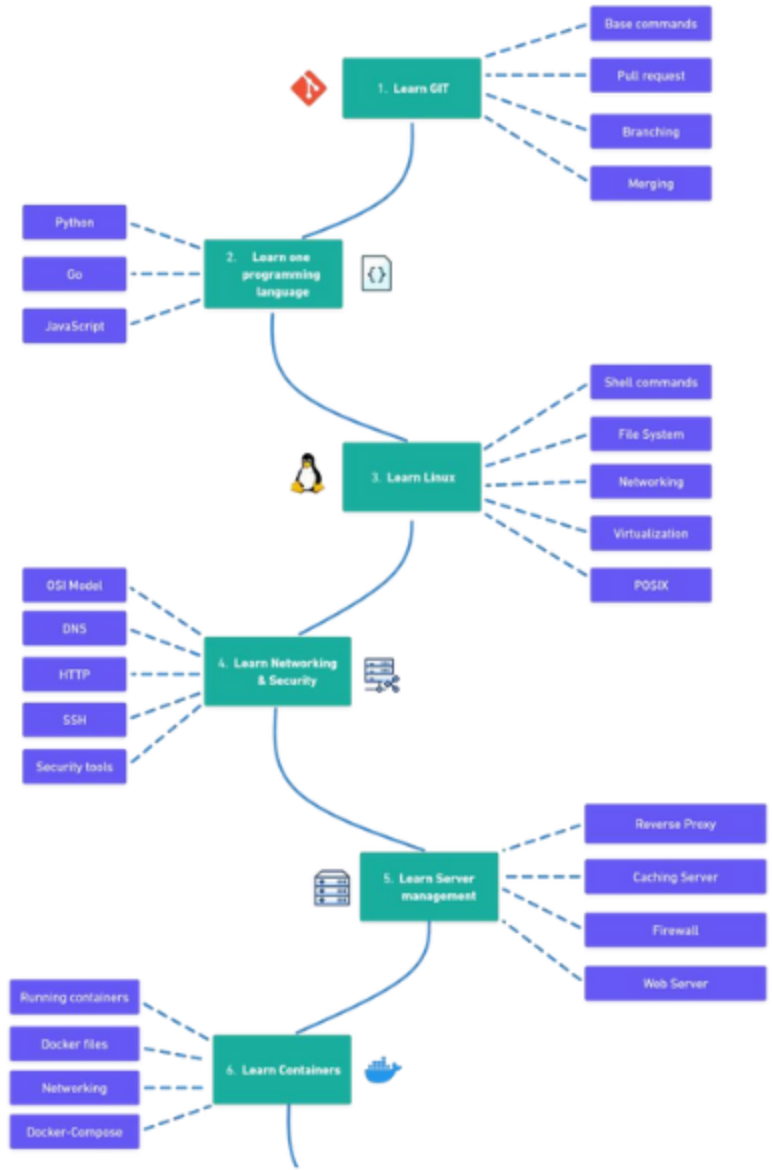
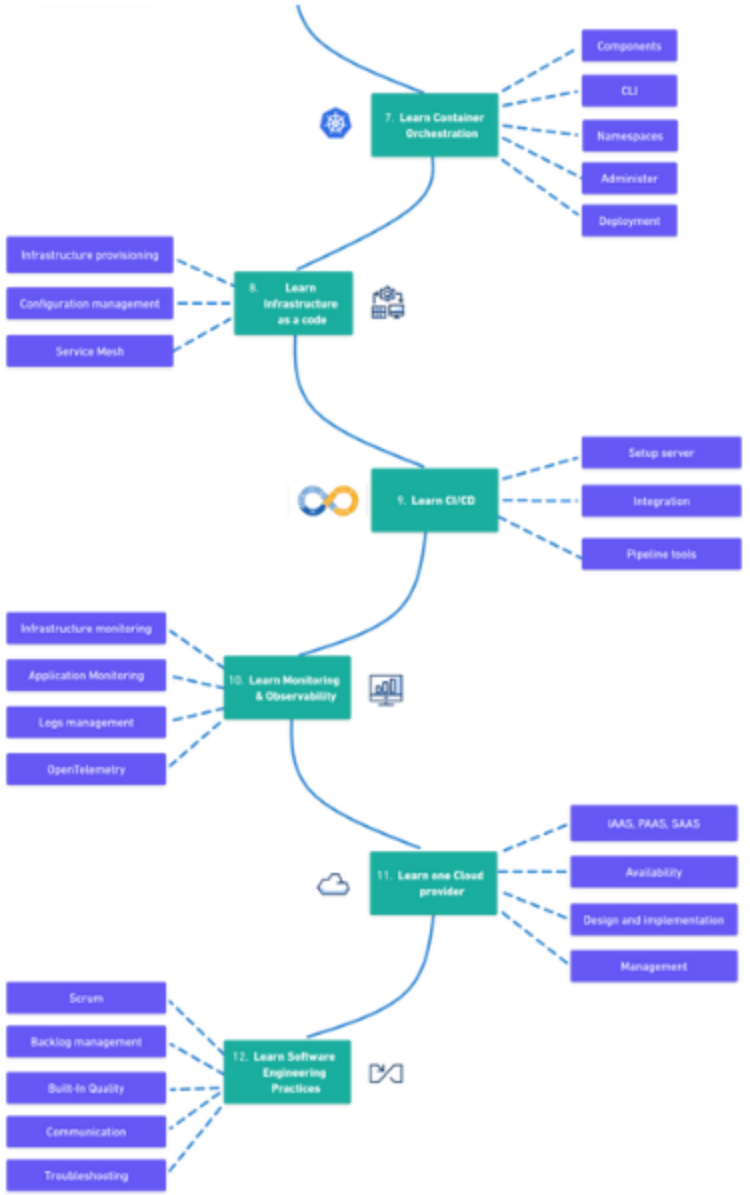
DevOps Curriculum

With Learning resources

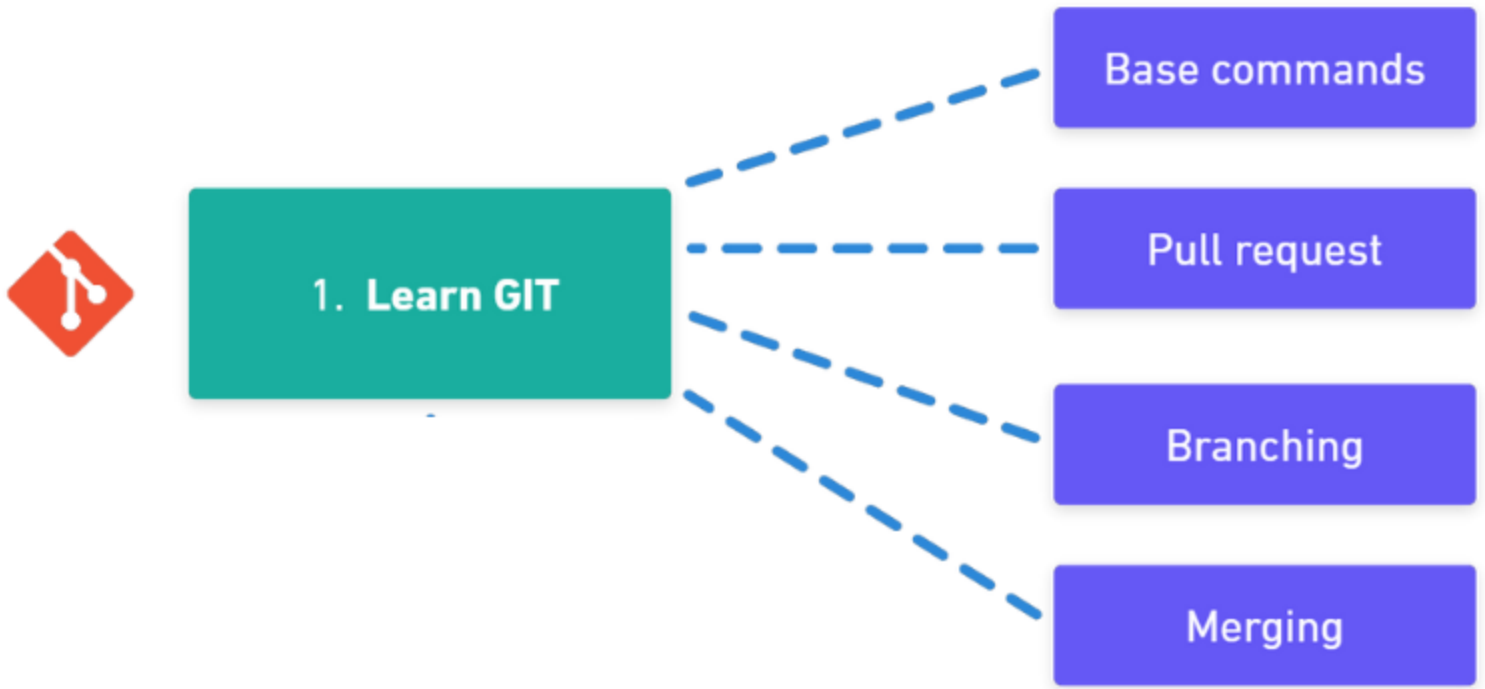
DevOps Curriculum



DevOps Curriculum



1. Learn Git



All your resources (files) will be held in a GIT repository. Those files are **application code** but also **infrastructure as a code**.

Gitis a free tool used for source code management. Git is used to tracking changes in the source code, enabling multiple developers to work together on non-linear development.

Two most popular Git platforms are **GitLab**and **GitHub**.

Here you need to learn Git commands, like git clone, branch, merge and how to collaborate on a project with pull requests.

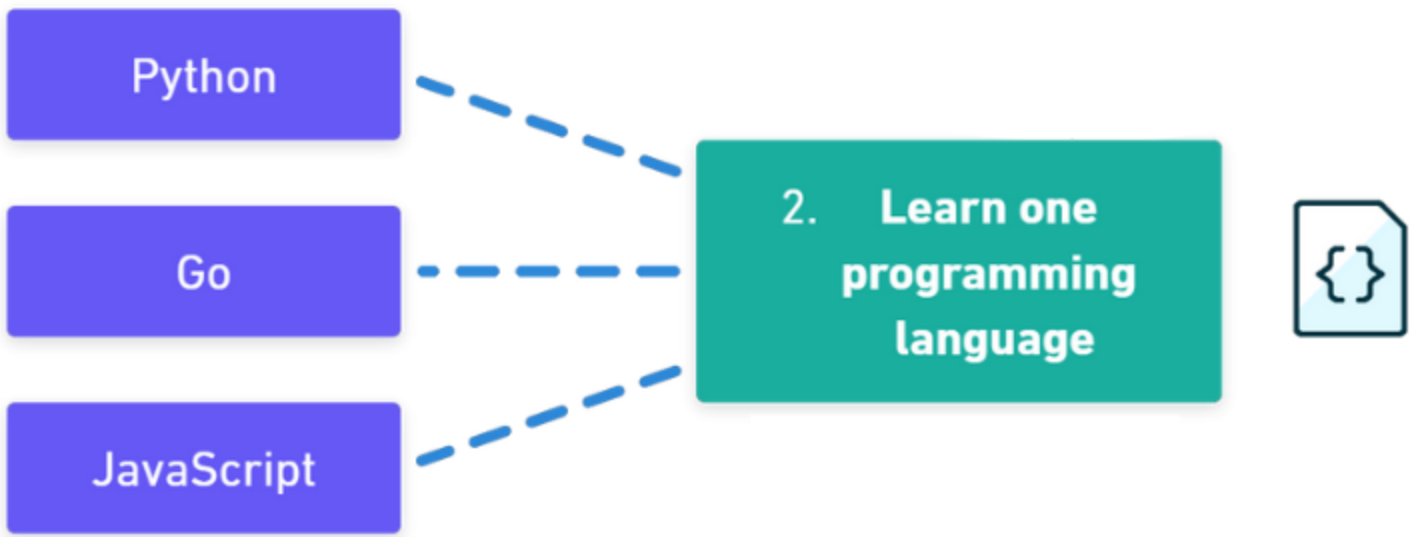
Learning resources

- https://www.atlassian.com/git
- https://learngitbranching.js.org/
- https://www.codecademy.com/learn/learn-git

GitLab

GitHub

2. Learn one programming language



As an engineer it is recommended to know at least one programming language that you can use to write **automation scripts**.

Some popular programming languages for DevOps-es are **Python, Go and JavaScript**.

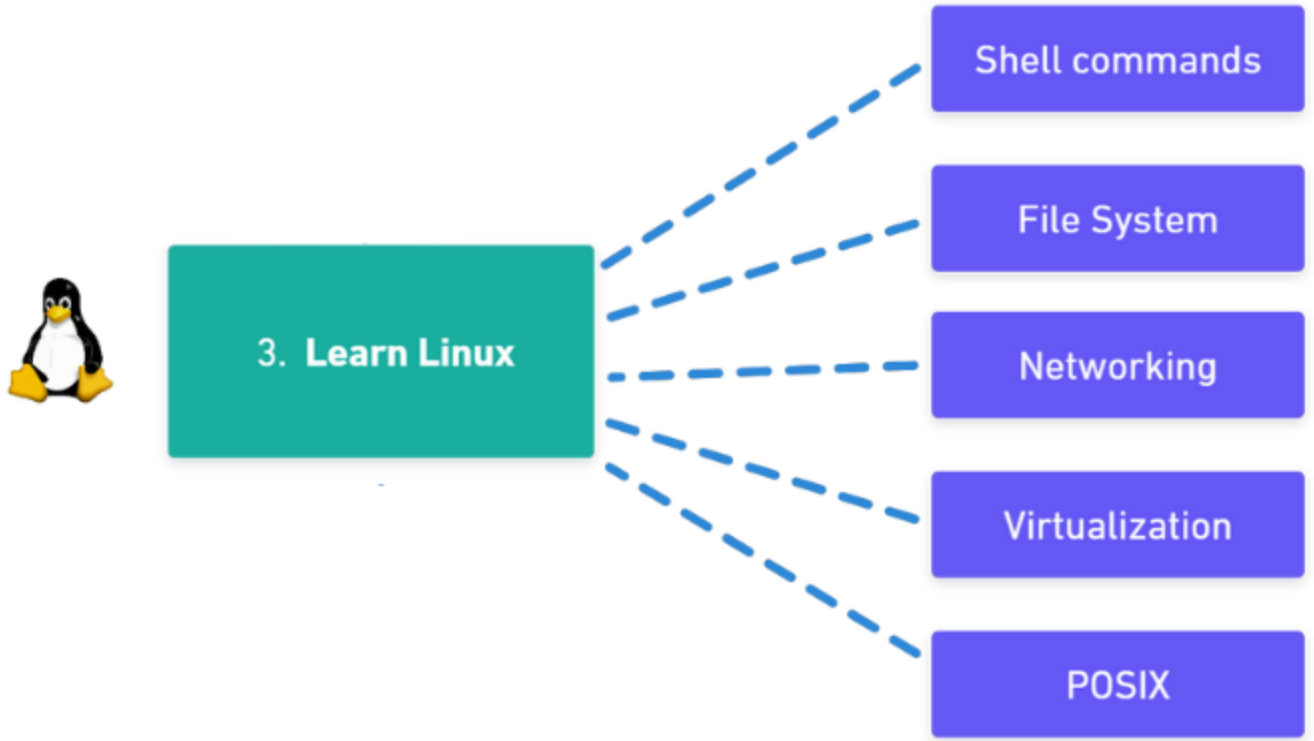
Python is a multi-paradigm language. Being an interpreted language, code is executed as soon as it is written, and the syntax allows for writing code in different ways. **Python**is frequently recommended as the first language new coders should learn, because of its focus on readability, consistency, and ease of use.

Here you need to learn basic concepts of programming languages, such as syntax, if/else, loops, data structures, etc.

Learning resources

- https://automatetheboringstuff.com/
- https://eloquentjavascript.net/
- https://ehmatthes.github.io/pcc/

3. Learn Linux & Scripting



An Operating system serves as a bridge between a computer's user and its hardware. Its function is to offer a setting in which a user can conveniently and effectively run programs.

As most servers use **Linux OS**, you need to make yourself comfortable with Linux and its CLI.

One easy to start distribution is **Ubuntu**.

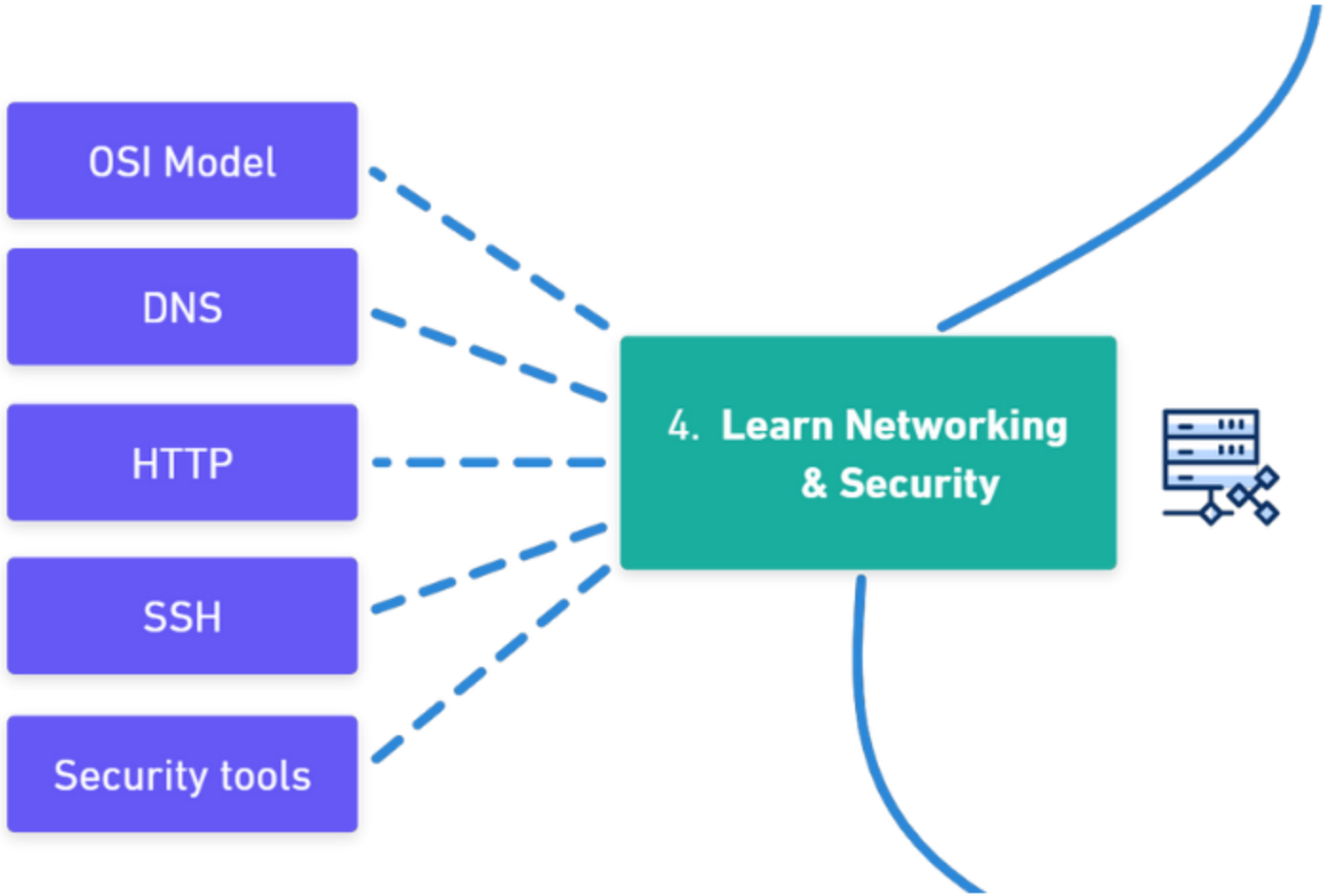
In addition, you need to know **scripting**to automate tasks for development and operations.

Here you can learn OS-specific languages, such as **Bash or Powershell**or independent, like Python or Go.

Learning resources

- https://www.tutorialspoint.com/operating_system/os_overview.htm
- https://www.shellscript.sh/
- https://www.guru99.com/powershell-tutorial.html

4. Learn Networking & Security



A **network protocol** is an established set of rules that determine how data is transmitted between different devices in the same network.

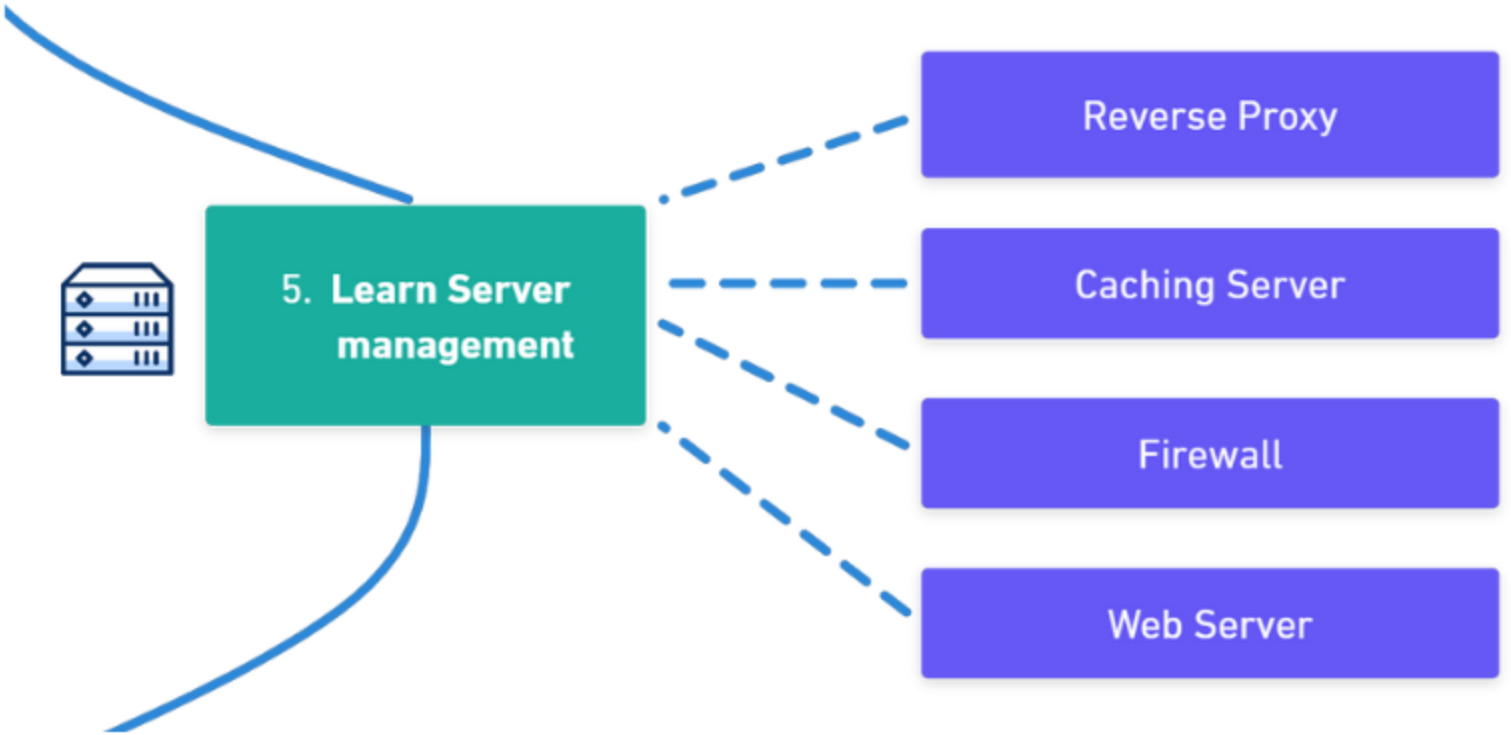
Essentially, it allows connected devices to communicate with each other, regardless of any differences in their internal processes, structure or design.

Here you will need to know how network works, how to configure **firewalls**, understand how **DNS**works, **OSImodel**, ipaddresses, ports, etc.

Learning resources

- https://www.youtube.com/watch?v=dV8mjZd1OtU
- https://www.amazon.com/Computer-Networking-Top-Down-Approach-7th/dp/0133594149
- https://www.pluralsight.com/courses/tcpip-networking-it-pros
- https://www.udemy.com/course/devsecops/

5. Learn Server Management



Server management includes all the infrastructure monitoring and maintenance required for servers to operate reliably and at optimal performance levels. The primary goals of an **effective server management** strategy are to:

- Minimize server slowdowns and downtime while maximizing reliability.
- Build secure server environments.
- Scale servers and related operations to meet the needs of the organization over time.

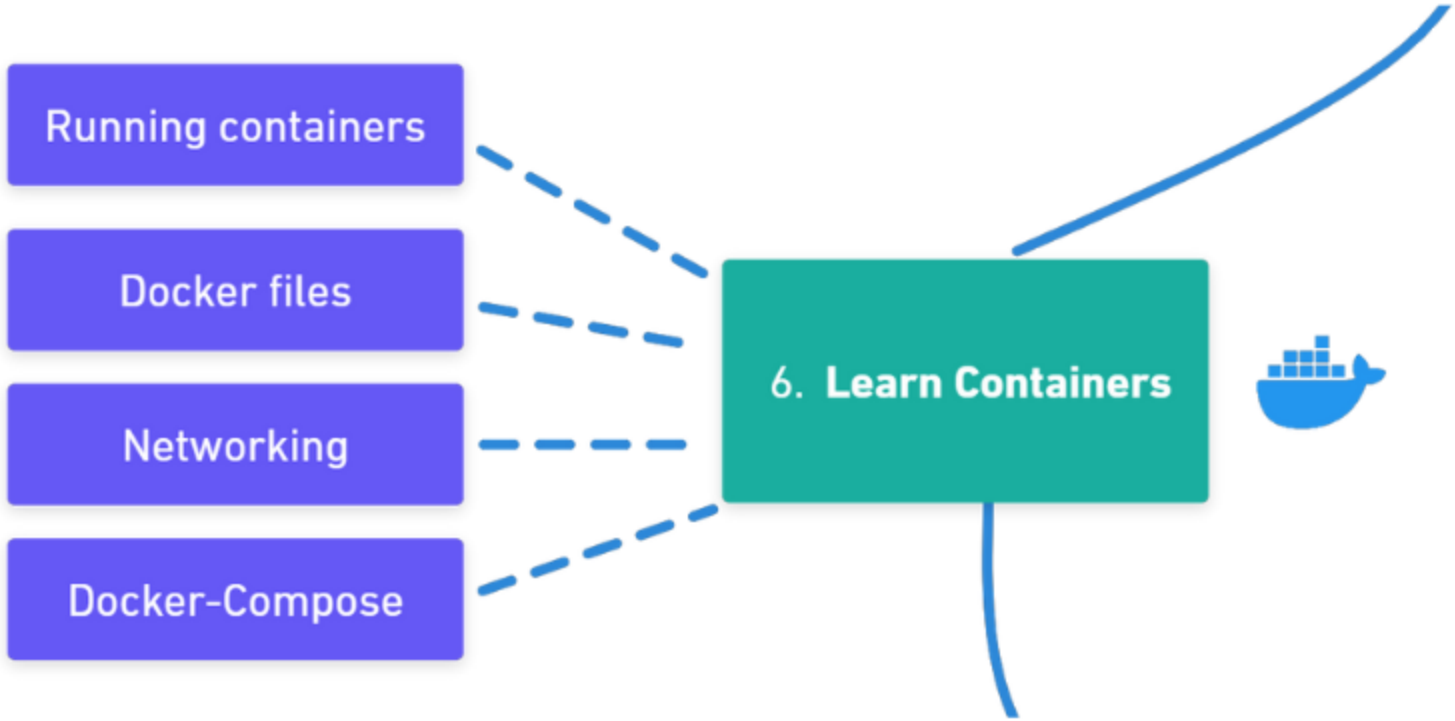
Here you will need to know what are **forward and reverse proxies**, **caching servers**, how to operate **Web Servers**, such as Nginx, Apache or IIS.

Learning resources

- https://www.cloudflare.com/en-gb/learning/cdn/glossary/reverse-proxy/
- https://www.cloudflare.com/en-gb/learning/performance/what-is-load-balancing/
- https://www.freecodecamp.org/news/the-nginx-handbook/

NGINX APACHE HTTP SERVER PROJECT

6. Learn Containers



A **container** is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another.

Docker is by far the most popular container technology today.

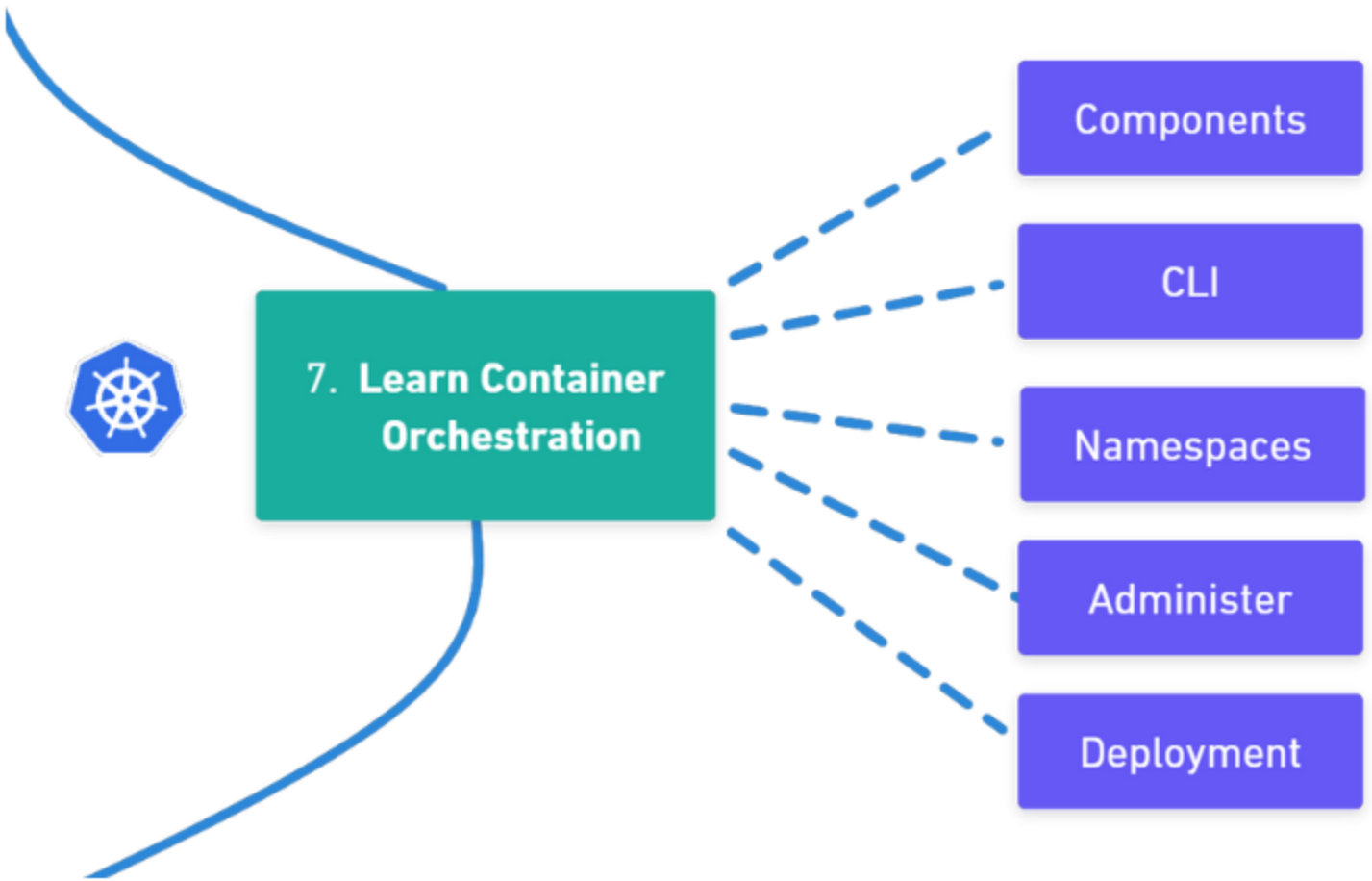
A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.

Here you need to know how to run containers, Docker Networking, Volumes, Dockerfiles, run multiple containers with Docker-Compose.

Learning resources

- https://cloud.google.com/learn/what-are-containers
- https://iximiuz.com/en/posts/container-learning-path/
- https://www.youtube.com/watch?v=3c-iBn73dDE

7. Learn Container Orchestration



Container orchestration **automates**the deployment, management, scaling, and networking of containers.

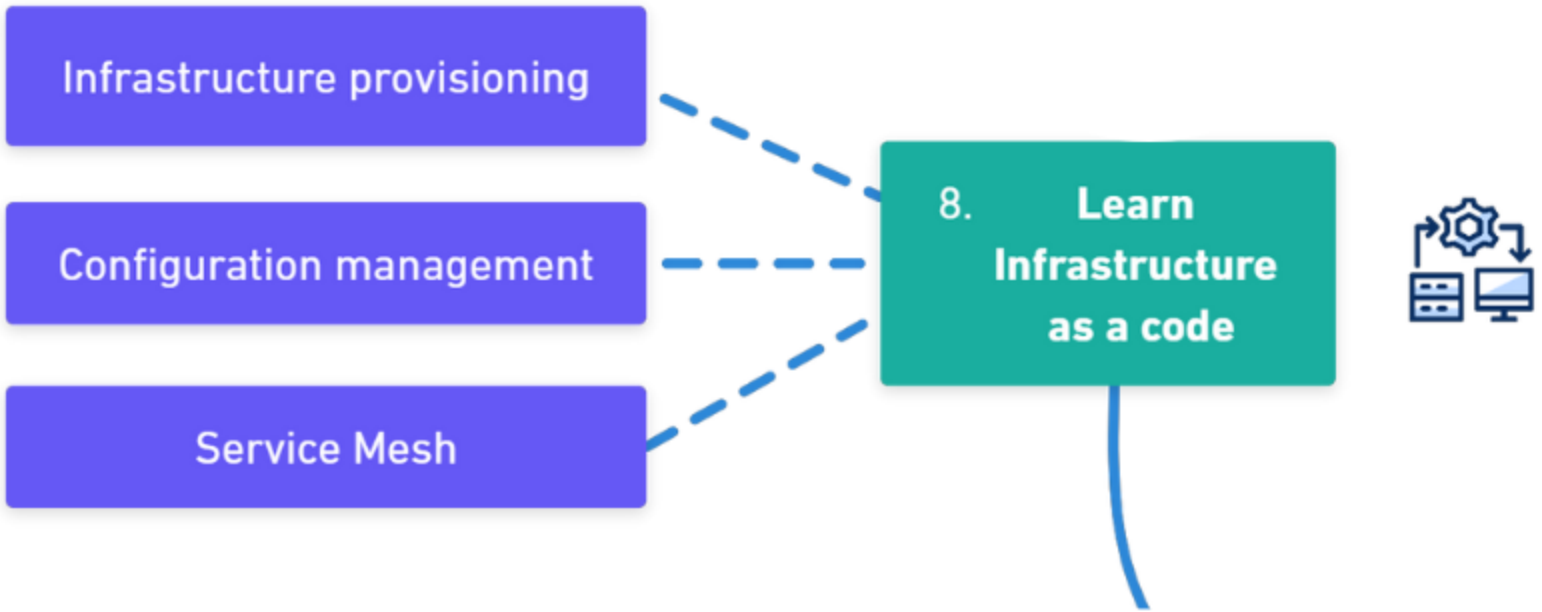
Container orchestration can be used in any environment where you use containers. It can help you to deploy the same application across different environments without needing to redesign it. And microservices in containers make it easier to orchestrate services, including storage, networking, and security.

Here you need to learn how **Kubernetes**works, how to administer Kubernetes cluster and deploy applications on it.

Learning resources

- https://www.youtube.com/watch?v=s_o8dwzRlu4
- https://thenewstack.io/primer-how-kubernetes-came-to-be-what-it-is-and-why-you-should-care/
- https://kodekloud.com/learning-path-kubernetes/

8. Learn Infrastructure as a code



Sometimes referred to as **IaC**, it refers to the techniques and tools used to define infrastructure, typically in a markup language like YAML or JSON. Infrastructure as code allows Engineers to automates environment setup and teardown. Accelerates and de-risks deployment by provisioning gold copy environments on demand.

Terraformis the most popular infrastructure provisioning tool.

Here you need to know how to do **infrastructure provisioning**and **configuration management**.

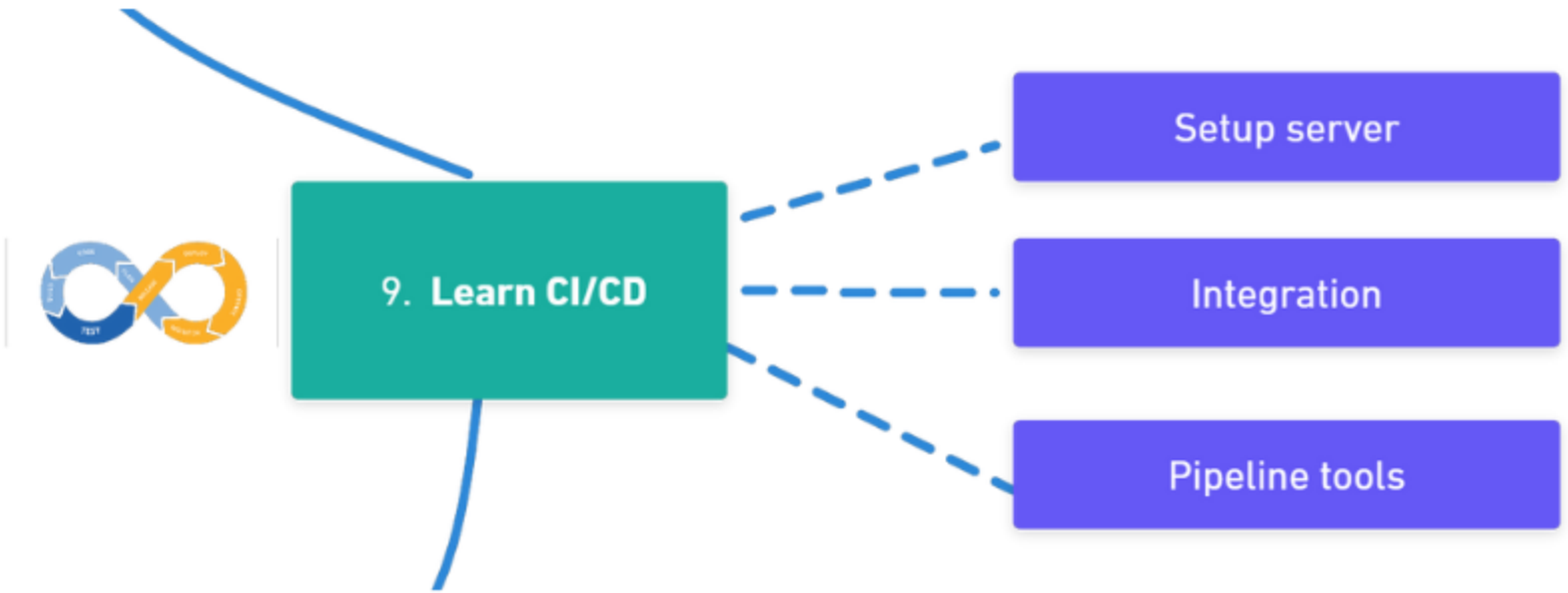
Learning resources

- https://thenewstack.io/guis-cli-apis-learn-basic-terms-of-infrastructure-as-code/
- https://blog.gruntwork.io/a-comprehensive-guide-to-terraform-b3d32832baca

HashiCorp Terraform

ANSIBLE

9. Learn CI/CD



Continuous Integration / Continuous Deployment is a method to frequently deliver apps to customers by introducing **automation** into the stages of app development. CI/CD is a solution to the problems integrating new code can cause for development and operations teams.

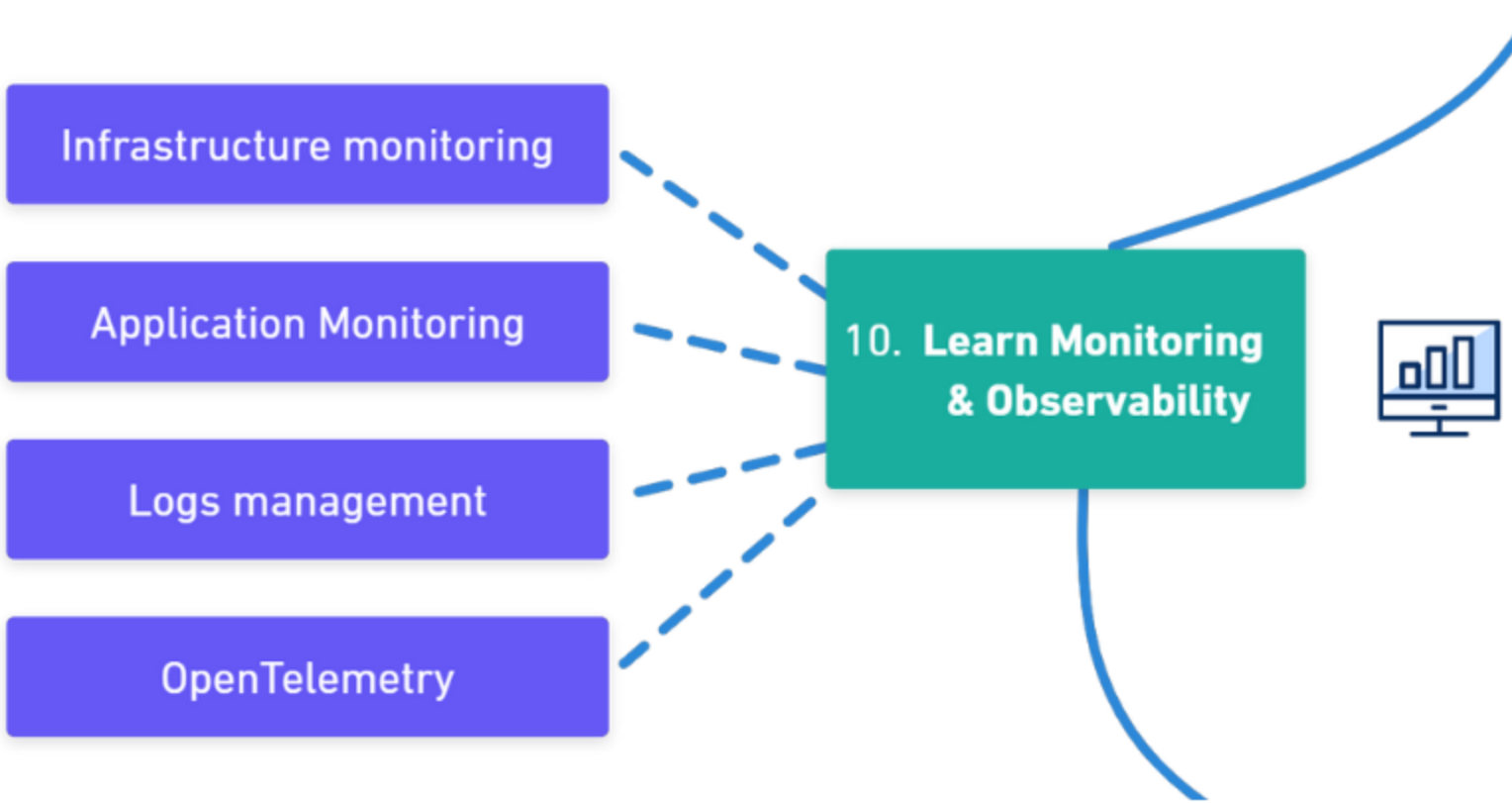
CI/CD introduces continuous automation and **continuous**monitoring throughout the lifecycle of apps, from integration and testing phases to delivery and deployment. These connected practices are often referred to as a "**CI/CD pipeline**" and are supported by development and operations teams.

Here you need to learn how to setup CI/CD server, integrate code and trigger pipelines automatically, build and package manager tools.

Learning resources

- https://semaphoreci.com/blog/cicd-pipeline
- https://www.udemy.com/course/jenkins-from-zero-to-hero
- https://docs.gitlab.com/ee/tutorials/

10. Learn Monitoring & Observability



Monitoring entails overseeing the entire development process from planning, development, integration and testing, deployment, and operations. It involves a complete and **real-time view** of the status of applications, services, and infrastructure in the production environment.

This is especially important when our software is in **production**, and we need to track all kinds of issues in our infrastructure and application.

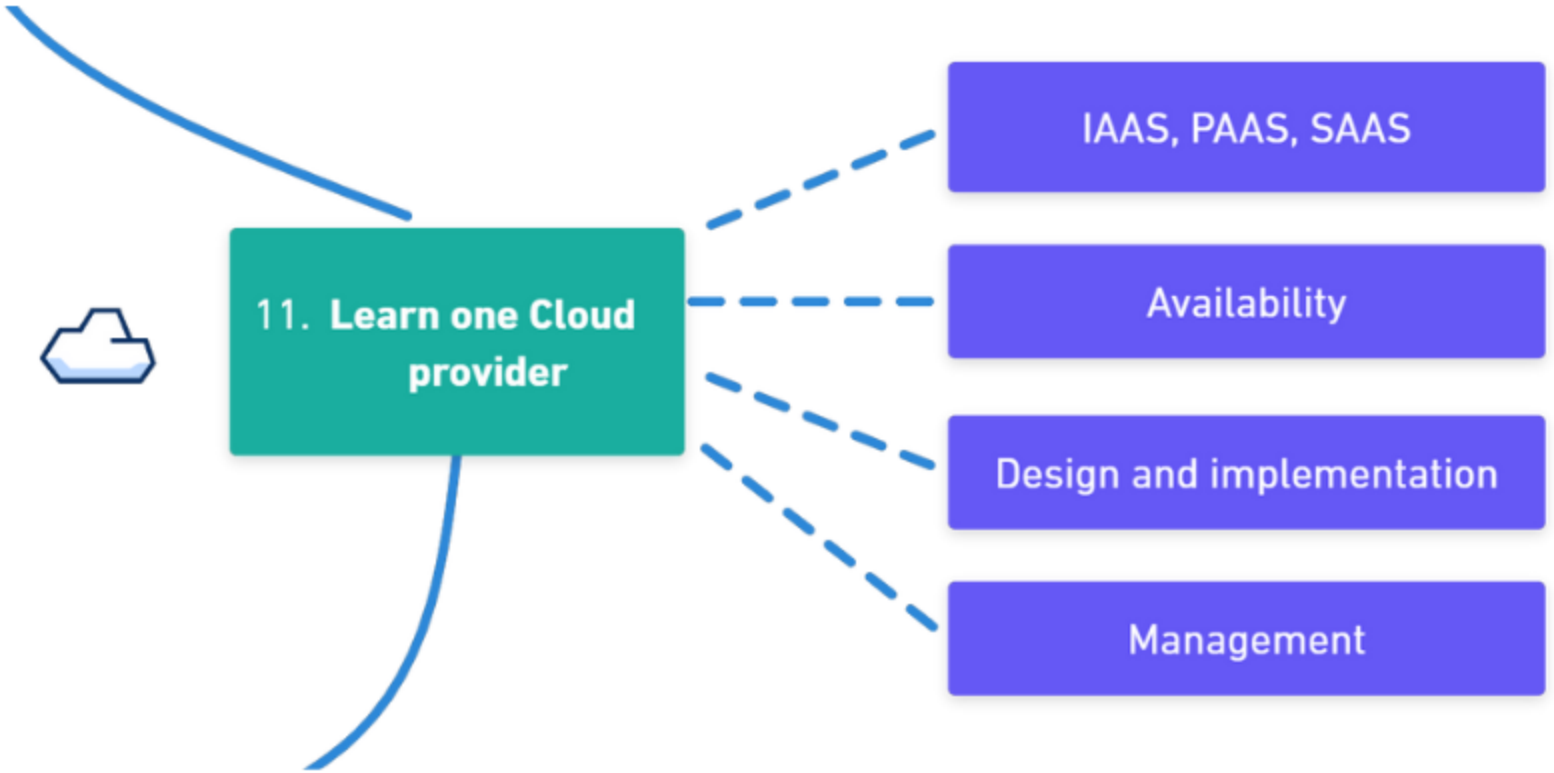
Two most popular tools are **Prometheus**and **Grafana**.

Here you need to know how to setup monitoring and visualize data.

Learning resources

- https://devopscube.com/what-is-observability/
- https://www.atlassian.com/devops/devops-tools/devops-monitoring
- https://thenewstack.io/applying-basic-vs-advanced-monitoring-techniques/

10. Learn one Cloud provider



Cloud providers provide a layer of APIs to abstract infrastructure and provision it based on security and billing boundaries. The cloud runs on servers **in data centers**, but the abstractions cleverly give the appearance of interacting with a single "platform" or large application. The ability to quickly provision, configure and secure resources with cloud providers has been key to both the tremendous success, and complexity, of modern DevOps.

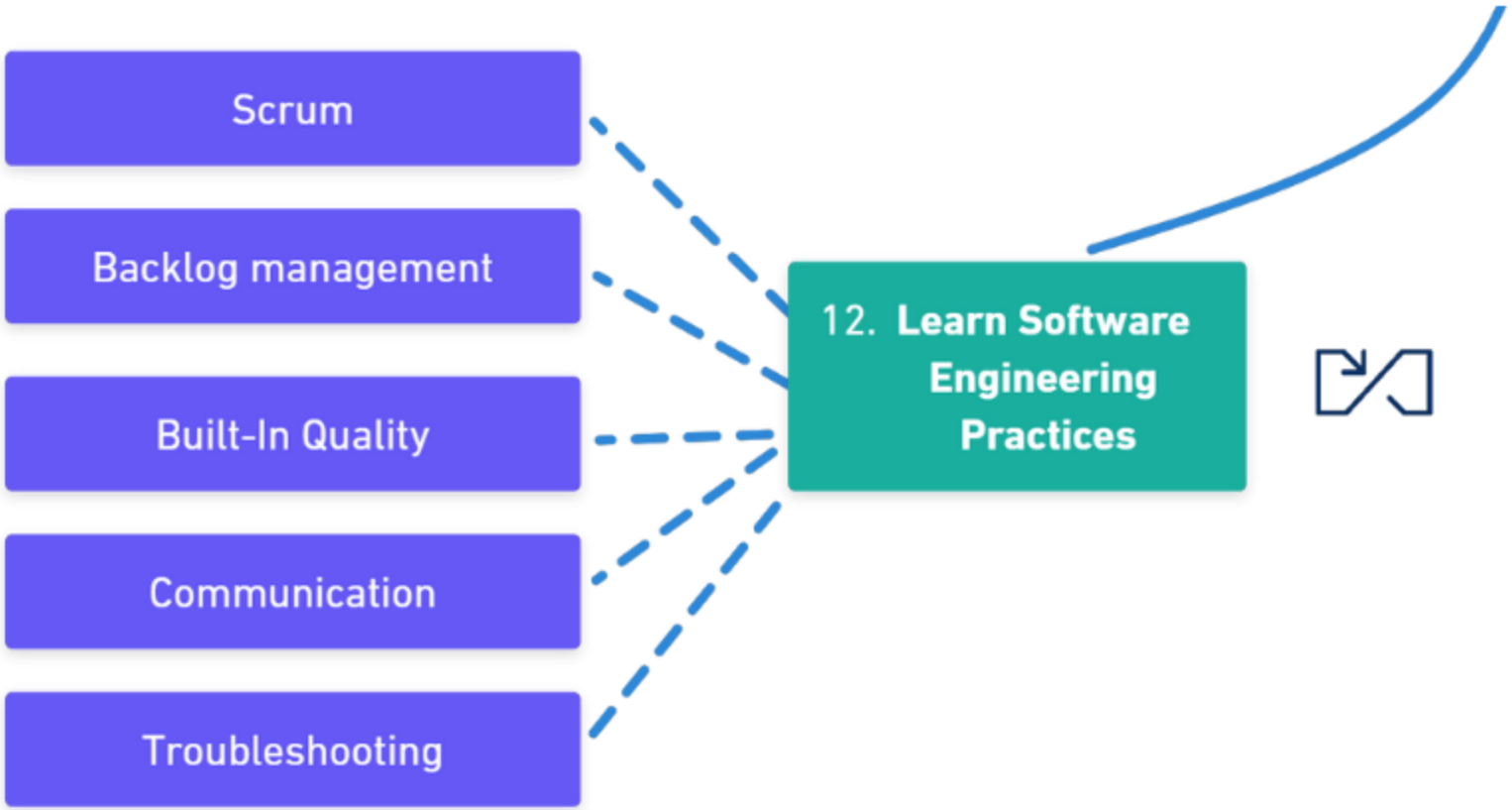
Most popular cloud providers in the market are **AWS**and **Azure**, as well as **Google Cloud**.

Here you need to know how to manage users and administration, networks, virtual servers, etc.

Learning resources

- https://learn.microsoft.com/en-us/certifications/exams/az-900
- https://www.udemy.com/course/aws-certified-cloud-practitioner-new
- https://acloudguru.com/learning-paths/aws-developer

11. Learn Soft. Engineering Practices



As a DevOps engineer you will probably work in the team with other developers in an Agile world, such as **Scrum**. So, it is very important to know different parts of **SDLC**, as well as tools which are used there.

In addition, it would be good to know how **automation testing** is working, as you will need to setup it in CI/CD way.

Here you need to know what is **Scrum**, all phases of **SDLC**, how **automation testing** is working etc.

Learning resources

- https://www.scrum.org/resources/ways-learn-about-scrum
- https://www.guru99.com/software-development-life-cycle-tutorial.html
- https://blog.testproject.io/2020/03/26/automation-testing-for-beginners-ultimate-guide/

DevOps Curriculum

Check the full list of learning resources in the repo

https://github.com/igitsanjeev/DevOps_Curriculum