12 Factor Applications and its security

By : Bilal Mazhar

About Me

Hi there, my name is Bilal and I will Welcome you to DevOps boot camp! I am thrilled to have you join us for this exciting journey of learning and discovery.

In this boot camp, we will be exploring the principles and practices of DevOps, which is a set of methodologies and tools that aims to bridge the gap between software development and operations. DevOps is an increasingly important area in the field of software engineering, as it helps organizations to streamline their processes, improve their agility, and deliver better value to their customers.

By the end of this boot camp, you will have gained a comprehensive understanding of DevOps and its key concepts, as well as practical skills in areas such as infrastructure automation, continuous integration and delivery, monitoring and logging, and more. You will be equipped with the knowledge and tools to apply DevOps principles in your own work and contribute to the success of your organization.

I am always looking to connect with other professionals in the field, share ideas and insights, and stay up to date on the latest trends and developments. I welcome the opportunity to connect with you and explore ways in which we can collaborate and support each other.

GitHub : https://github.com/BilalMaz/DevOps-Architect-BootCamp
LinkedIn : https://www.linkedin.com/in/bilalmazhar-cyber-security-consultant/
Please find my LinkedIn profile

What is 12 factor application

The 12 factor application is a **methodology** for building **modern, cloud-native applications** that are **scalable**, **resilient**, and easy to maintain. The methodology was originally introduced by developers at Heroku in 2011, and has since become a widely adopted standard for building cloud-native applications.

https://12factor.net/

Codebase

→ A **single codebase** is used for each application, which is tracked in a **version control system**. Multiple deploys can be made from the same codebase, and each deploy should be a **separate release**, identified by a **unique release ID**. This ensures that everyone is working off of the same codebase and there is no divergence between **development**, **staging**, and **production**.

→ Security : **Code reviews** and **audits** should be performed to ensure the codebase is secure and free from vulnerabilities. Access to the codebase should be limited to **authorized** personnel only.

Security	Tools
Static Analysis	SonarQube , Burp Suite , Nessus , Checkmarx , CodeSona , Semgrep
Dynamic Analysis	OWASP ZAP , OpenVAS , WebInspect , Nikto , Arachni , Vega

Dependencies

→ Explicitly **declare** and **isolate dependencies**. Dependencies should be explicitly declared and managed through a **package manager**. Isolation should be achieved by using **virtualized environments** such as **Docker containers**, which provide a consistent and reproducible runtime environment for the application.

→ Security : Dependencies should be kept **up-to-date** and **regularly patched** to address security vulnerabilities. Vulnerability scanning tools should be used to detect and mitigate any potential risks.

Security	Tools
Dependency checkers	OWASP Dependency-Check , Retire.js, Yarn Audi , Safety, Dependency-Track

Config

→ Store configuration in the environment. Configuration should be stored in **environment variables**, which can be **easily managed** and changed without **affecting the application code**. This allows for easy and **flexible configuration management** across different environments.

→ Security : Sensitive configuration data such as **API keys** or **database passwords** should be **encrypted** and **stored securely**. Access to configuration data should be **limited** to authorized personnel only.

Security	Tools
Secure Configuration	HashiCorp Vault, Ansible , Terraform, Conftest , CyberArk Conjur,

Backing services

→ Treat backing services as attached resources. Backing services such as **databases**, **message queues**, and **caches** should be treated as attached resources that can be easily **added** or **removed** without **affecting** the application code.

→ Security : Access to backing services should be secured using **strong authentication mechanisms** such as SSL/TLS or SSH. **Authorization mechanisms** should be used to **restrict access** to sensitive data

Security	Tools
Securing Backing services	DbShield , Zscaler , Duo Security , OSSEC , Nessus, Nexpose

Build, Release, Run

→ Strictly separate build and run stages. The build stage should be separate from the run stage, and the release stage should be separate from both. This ensures that each stage can be tested and verified independently, and that the same artifact can be deployed to any environment.

→ Security : Secure build systems should be used to prevent unauthorized access to the build environment or artifacts. Release management processes should be used to ensure only authorized personnel can deploy to production environments.

Security	Tools
Secure CI / CD	Jenkins , Gitlab , Docker Hub , GitHub Actions

Processes

→ Execute the app as one or more **stateless processes**. The application should be designed to run as one or more **stateless processes**, which can be easily scaled out horizontally. Stateful components such as databases should be treated as **backing services**.

→ Security : Securely **isolate processes** and use **containers** for added security and scalability

Security	Tools
Secure Process	Docker, Kubernetes,Amazon ECS , Google Kubernetes Engine (GKE)

Port Binding

→ Port Binding refers to the **process of attaching a process** or an application to a **specific network port** on a computer. This allows the process or application to **receive incoming network traffic** on that **particular port**, and send **outgoing traffic from that port**

→ Security : Securely expose services using Transport Layer Security (TLS) and secure ports

Security	Tools
Secure Network	Let's Encrypt, Certbot, HashiCorp Vault, OpenSSL , Nmap , Firewalls , SEIM

Concurrency

→ Concurrency refers to the ability of an application to execute multiple tasks or processes at the same time, independently of each other. This means that multiple tasks can be performed simultaneously within the same application, without interfering with one another.

→ Security : Securely manage and monitor concurrent processes and use load balancers for traffic routing and failover

Security	Tools
Securing concurrency	Kubernetes, HashiCorp Nomad, Amazon Elastic Load Balancer, HAProxy

Disposability

→ Disposability refers to the ability of an application to **start up** and **shut down quickly** and **gracefully**. This means that an application can be **easily started or stopped** without **affecting the availability** or performance of other applications or services running on the same system

→ Security : Improve the application's **resilience** by ensuring that it can be started up and shut down quickly and gracefully.

Security	Tools
Securing Disposale	Process managers , Orchestration , Load balancers , Logging and monitoring

Dev/prod parity

→ Keep **development**, **staging**, and **production** as similar as possible

→ Security : Securely **maintain identical environments** across the software development lifecycle (SDLC) and use infrastructure as code (IaC) for consistency

Security	Tools
Securing concurrency	Process managers , Orchestration , Load balancers , Logging and monitoring

Logs

→ Logs refer to the **records generated by an application** or system that document its **activities** and events. These records can include information about user actions, system errors, resource usage, and more. Logs can be used for a variety of **purposes**, including **troubleshooting issues**, **auditing activities**, and **analyzing performance**

→ Security : Securely **collect** and **aggregate logs** and use **centralized logging** for easier analysis

Security	Tools
Logging and monitoring	ELK Stack, Splunk, Graylog, Fluentd , Splunk , Qradar , Wuzul

Admin processes

→ Admin processes refer to the tasks and actions performed by administrators in order to manage and maintain an application or system. These tasks can include activities such as installing updates, configuring settings, and monitoring performance

→ Security : Securely automate one-time admin tasks and implement strong access controls and auditing

Security	Tools
Secure Admin acces to to process	Kubernetes CronJobs, HashiCorp Nomad, Jenkins, AWS Lambda