CHEAT **Essential Development Tools & Concepts Cheatsheet** SHEETS

A concise reference to key tools and concepts in modern software development, aiding in efficiency and best practices.

Version Control with Git

Basic Git Commands

git init	Initializes a new Git repository.
git clone <url></url>	Clones a repository from a URL.
git add <file></file>	Adds a file to the staging area.
git commit -m " <message>"</message>	Commits changes with a descriptive message.
git push origin <branch></branch>	Pushes local commits to a remote repository.
git pull origin <branch></branch>	Pulls changes from a remote repository.
git status	Shows the status of the working directory.
git branch	Lists all local branches.
git checkout <branch></branch>	Switches to the specified branch.

Branching and Merging

git branch <new-branch></new-branch>	Creates a new branch.
git checkout - b <new-branch></new-branch>	Creates and switches to a new branch.
git merge <branch></branch>	Merges the specified branch into the current branch.
git log	Shows the commit history.
git diff	Shows changes between commits, branches, etc.

Undoing Changes

git revert <commit></commit>	Creates a new commit that undoes the changes made in the specified commit.
git reset HEAD <file></file>	Unstages a file from the staging area.
git checkout - - <file></file>	Discards changes in the working directory.

Containerization with Docker

Basic Docker Commands

<pre>docker build -t <image-name> .</image-name></pre>	Builds a Docker image from a Dockerfile.
docker run <image-name></image-name>	Runs a container from an image.
docker ps	Lists running containers.
<pre>docker stop <container-id></container-id></pre>	Stops a running container.
docker rm <container-id></container-id>	Removes a stopped container.
docker images	Lists all available Docker images.
docker rmi <image-id></image-id>	Removes a Docker image.

Docker Compose

docker - compose up	Builds, (re)creates, starts, and attaches to containers defined in a docker-compose.yml file.
docker - compose down	Stops and removes containers, networks, volumes, and images defined in a docker-compose.yml file.
docker - compose ps	Lists the containers managed by Docker Compose.

Dockerfile Instructions

FROM <image/>	Specifies the base image for the Docker image.
RUN <command< td=""><td>Executes a command during the image build process.</td></command<>	Executes a command during the image build process.
COPY <src> <dest></dest></src>	Copies files or directories from the host to the container.
WORKDIR <path></path>	Sets the working directory for subsequent instructions.
EXPOSE <port></port>	Exposes a port for network traffic.
CMD <command< td=""><td>Specifies the default command to run when the container starts.</td></command<>	Specifies the default command to run when the container starts.



Continuous Integration/Continuous Deployment (CI/CD)

Key Concepts

Continuous Integration (CI): Automates the integration of code changes from multiple developers into a shared repository. It involves automated testing to detect integration errors early.

Continuous Deployment (CD): Automates the release of code changes to production or staging environments. It extends CI by automatically deploying all code changes that pass the automated tests.

Continuous Delivery: Similar to Continuous Deployment, but requires manual approval for deployment to production. Automates the steps up to the production deployment.

Jenkins	An open-source automation server that supports building, testing, and deploying software.
GitLab Cl	Integrated CI/CD pipeline within GitLab for automated building, testing, and deployment.
GitHub Actions	Automates software workflows directly in your GitHub repository.
CircleCl	A cloud-based CI/CD platform that automates the build, test, and deployment process.
Travis Cl	A cloud-based CI service used for building and testing software projects hosted on GitHub and Bitbucket.

Common CI/CD Tools

Pipeline Stages

Typical CI/CD pipelines include stages like:

- Build: Compile code and create artifacts.
- **Test**: Run automated tests (unit, integration, end-to-end).
- Package: Bundle artifacts for deployment.
- **Deploy**: Deploy artifacts to the target environment (staging, production).

Code Quality and Linters

Code Quality Metrics

Code Coverage: Percentage of code executed by tests.

- **Cyclomatic Complexity**: Measures the complexity of a program by counting the number of linearly independent paths through the source code.
- Maintainability Index: Indicates the ease with which software can be maintained.

ESLint	A linter for JavaScript and JSX.
Stylelint	A linter for CSS and SCSS.
SonarQube	A platform for continuous inspection of code quality.
PMD	A source code analyzer for Java, JavaScript, and other languages.
Checkstyle	A tool for checking Java code style.

Linters and Code Analysis Tools

Benefits of Using Linters

- Enforces consistent coding style across the project.
- Detects potential bugs and errors early.
- Improves code readability and maintainability.
- Reduces code review time.