Connect to the PROJECT_VM

Overview

For this project, you will use established Microservices techniques to design and implement an end-to-end prototype for the application described below. The expectation is that you and your team will be able to demonstrate the application and lead a final presentation of your findings on day 8.

Problem Definition

“Metropolitan Convention Center” would like to create a web site where participants can sign-up and buy tickets for events held in the center’s main hall and meeting rooms.

As a first step they are interested in seeing a prototype that allows users to:

- Create a login for the site
- Login to the site
- Look at a list of events
- Register for one or more events

Technical Requirements

The prototype’s architecture should include:

- A front end web application
- Multiple RESTful API Microservices

The prototype should be set up to run on an individual machine.

The Microservices should be set up to demonstrate the following capabilities:

- Build using Gradle
- Dockerization
- CI Pipeline
- Load balancing for the APIs
- Metrics capture
- Tracing

Development and testing environment setup

To allow the teams to succeed in the task at hand, each team should prepare a development environment that includes:

- Java
- Git
- NodeJS
- IDE for developing Java applications
- Gradle
- Docker
- Jenkins
- Kubernetes
- Prometheus
- Jaeger

It is suggested that these be installed in a Linux virtual machine. Further information is provided in the technologies list below.

Project Technologies List

The following technologies will be used in to fulfill project requirements.

Technology	Description	Purpose
Virtualization Platform	Virtual box, VMWare or similar environment	Host a virtual machine that can be used for development and deployment
Linux	Operating System	OS for the VM. Works well with other technologies (Docker, Kubernetes, etc…)
Git	Source code management	Manage source code locally and push to central repository
GitHub account	Central source code repository	Central repository
Java	Programming language	Supports Spring framework that will be used to create RESTful APIs
Spring Boot	Java application framework	Java code library for creation of standalone server applications and RESTful APIs
Eclipse	Integrated Development Environment	Supports source editing for Java applications
Gradle	Build Tool	Building Java applications
React	JavaScript framework	Used to create single page client applications (SPAs) that run inside the web browser
NodeJS	Programming platform and dependency tools	Supports the client application which is based in React and uses node based tools for build and development hosting
Postman	Generic HTTP client	Supports development & testing RESTful APIs
Docker	Virtual container system	Used to create and deploy light weight virtual application runtime environments
Jenkins	CI Pipeline/Build Manager	Manage multi stage builds
DockerHub account	Cloud-based Docker container image management	Allows posting and sharing of docker images
Nginx	Web server application	Serve React apps in Docker container
Kubernetes	Container orchestration system	Automates deployment, scaling and management of Docker containers
Prometheus	Event and alert monitoring system	Monitoring and alerting for server apps
Jaeger	Tracing system	Provides tracing support for server apps
JJWT	Java Lib for working w/JWT tokens	Provides token creation and validation

Daily Lecture/Labs/Project Schedule:

Students will work on the project each afternoon over eight days.

The daily schedule looks like this:

Times	Activity
8:30 am – 12pm	Course Lecture and Labs
12pm – 1pm	Lunch Break
1pm – 4pm	Project Work

Further information about what happens during Project Work time is given in the Daily Project Work sections below.

"A React framework based client is provided. Documentation for that can be found at the very end of this document"

Course Day 1: Sprint #1 (Day 1 of 2):

Schedule

Times	Activity
1pm – 1:10 pm	Instructor describes today's tasks and deliverables
1:10 pm – 4pm	Project work time, Instructor will come around to check on progress and provide pointers

Description

The instructor start the project session with a demonstration of the application you are going to be creating. They will then divide the class up into pairs. Each pair will work on the project together. You will then have time to work on the project. A virtual machine will be provided that includes all software required for project development. Your tasks for today revolve around getting familiar with the development environment and creation of design and work plans. Later in the work period the instructor will come around to each pair to discuss their findings and progress.

Tasks

- Create an application design that aligns with the fundamentals of DevOps and Microservices. The design should include a list of components and services along with information about what each will be used for and how they are expected to work together.
- Create a work plan that includes a backlog of tasks to be completed during development.
- Create a local Git repository in your development environment
- Create a GitHub online account and create a repo on it that links to your local repository.

Daily Milestones

By the end of the day your pair should have completed your application design and development plans. In addition you should have gotten familiar with the setup of the development VM and created a Git repository.

- Application design document
- Development plan document
- Local development environment
- Local Git repository
- GitHub repository

Course Day 2: Sprint #1 (Day 2 of 2)

Schedule

Times	Activity
1pm – 1:10 pm	Instructor describes today's tasks and deliverables
1:10 pm – 3pm	Project work time
3pm – 4pm	Pairs discusses what they've learned

Description

Extend your integration plan from session 1 by designing a build process using the Gradle build tool. Develop a functional RESTful service that can respond to queries and test the build process. The end result of Sprint one should be a build process and an operational Proof-of-concept RESTful service. Since this is the end of a sprint, the last hour will be reserved for the teams to discuss lessons learned.

Tasks

- Develop a RESTful service that accepts parameters and returns Customer data based on them.
- Design a build process using Gradle that will be used to build your APIs.
- Run the API and use a browser to check the functioning of the created service's GET endpoints
- Commit the completed API code and build.gradle file to your local repository
- Prepare to explain and demonstrate the service and build process

Sprint Deliverables

By the end of the day your pair should be ready to explain your build process and demonstrate an initial RESTful API.

- A functional RESTful API that serves hard coded Customer data.
- A functional Gradle build process

Course Day 3: Sprint #2 (Day 1 of 2)

Schedule

Times	Activity
1pm – 1:10 pm	Instructor describes today's tasks and deliverables
1:10 pm – 4pm	Project work time, Instructor will come around to check on progress and provide pointers

Description

Using the Spring Framework your pair will develop a state-full back-end API that works with the provided frontend application. The data being sent to the REST service will be stored in an internal SQL database. Your RESTful service at this point should be stateless and should be able to communicate with as well as store all persistent data in the database. The service, when completed, should be deployed in Docker and functionally tested.

Tasks

- Create a new local Git repository for the Spring Boot API project
- Implement the Customer API with full CRUD capabilities using Spring Boot
- The API should use data from an embedded database
- Use Gradle to build and run the Spring Boot API project
- Run the new API and check the endpoints using Postman Requests
- Commit the completed API code and build.gradle file to your local repository
- Hand-code a Dockerfile and use it to create a working Docker image.
- Commit the Dockerfile to Git
- Bring up the API using Docker
- Verify that the API functions correctly when run using Docker

Daily Milestones

By the end of the day your pair should be able to show and explain the completed RESTful API. You should also be able to show the API working when run in a Docker container.

- A functional Docker image/container for the Spring Boot API project
- A functional RESTful API based on Spring Boot that serves Customer data from an internal database.

Course Day 4: Sprint #2 (Day 2 of 2)

Schedule

Times	Activity
1pm – 1:10 pm	Instructor describes today's tasks and deliverables
1:10 pm – 2pm	Project work time
2pm – 4pm	Teams present what they have achieved this week

Description

Today you will create a new API that generates a JWT token for authentication and authorization of your existing Customer API. You will update the Customer API to deny any requests that do not include a valid token. You will develop integration tests using Postman to ensure that the Customer API is able to work properly with the Security API.

Tasks

- Create a new local Git repository for the Security API project
- Implement the Security API using Spring Boot and a JSON web token library.
- Use Gradle to build the Security API project
- Hand-code a Dockerfile and use it to create a working Docker image.
- Bring up the security API using Docker
- Run the security API and check its endpoints using Postman
- Try accessing the Customer API using a token provided by the Security API
- Create Postman test scripts that verify the integration between the two APIs
- Commit the completed API code and build.gradle file to your local repository
-

Deliverables

By the end of the day your pair should be able to show the security API working with the Customer API. Time permitting you should also be able to show the new APIs working with the provided front end application.

- A functional Docker image/container for the Security API project
- A functional RESTful API that checks credentials and returns JWT tokens.
- An updated Customer API that rejects any requests that don't include a valid token
- Postman scripts that test and verify the security API and its integration with the Customer API

Course Day 5: Sprint #3 (Day 1 of 2)

Schedule

Times	Activity
1pm – 1:10 pm	Instructor describes today's tasks and deliverables
1:10 pm – 4pm	Project work time, Instructor will come around to check on progress and provide pointers

Description

For this sprint you will deploy the application into a simulated production setup. The setup will include a service discovery configuration, load balancing and optionally metrics and tracing. Your will deploy the application data API and security API into separate load balancing groups. Postman test scripts should be created to ensure that the setup is running correctly and can survive individual instance failures. Your setup should provide for the service discovery and load balancing using Kubernetes.

Tasks

- Setup the Security and Customer API services to run with load balancing.
- Setup the front end to work with the load balanced addresses of the back end services.
- Create Postman test scripts to ensure the setup is working properly
- Backend API services should be running inside Docker containers.
- Demonstrate the front-end application running with the new setup.

Daily Milestones

By the end of the day you should have load balanced partially configured.

- Kubernetes(MiniKube) should be up and running
- MiniKube Dashboard should be functioning
- Docker images for the data and security services should be completed

Course Day 6: Sprint #3 (Day 2 of 2)

Schedule

Times	Activity
1pm – 1:10 pm	Instructor describes today's tasks and deliverables
1:10 pm – 3pm	Project work time
3pm – 4pm	Pairs discusses what they've learned

Description

Today you will continue the work started yesterday - setting up your application with a service discovery configuration and load balancing. If time permits you should introduce metrics and tracing along with manual tests that verify their functionality.

Tasks

- Continue tasks from yesterday.

Optional Tasks

If time permits complete these tasks as well:

- Setup an API to generate metrics using Prometheus
- Setup an API to trace requests using Jaeger

Deliverables

By the end of the day you should be ready to demonstrate the provided front end application working with the new load balanced back end API setup.

- A functional load balancing setup for the two APIs
- The front end app configured to access the APIs through the load balancers.
- Postman test scripts to verify the new setup

Course Day 7: Sprint #4 (Day 1 of 2)

Schedule

Times	Activity
1pm – 1:10 pm	Instructor describes today's tasks and deliverables
1:10 pm – 4pm	Project work time, Instructor will come around to check on progress and provide pointers

Description

The projects VMs include a Jenkins server that each pair should use to develop a CI pipeline that automates the build, testing and dockerization of their application. Following the creation of a CI pipeline, the team will extend their previous work to provide continuous delivery. As part of the CD pipeline, a test environment should be created using Docker to perform integration and system testing. By the end of the iteration, a full CD pipeline is demonstrated.

Tasks

- Get the Jenkins server up and running
- Create a CI Pipeline that builds the API server
- Add to the pipeline the ability to create a Docker image for the API

Deliverables

By the end of the day your pair should have a working CI pipeline that builds and dockerizes your API. In addition you should run your previously created Postman test scripts against the Docker API image created by the pipeline.

- A functional Jenkins server
- A functional CI pipeline
- Generated Docker images

Course Day 8: Sprint #4 (Day 2 of 2)

Schedule

Times	Activity
1pm – 1:10 pm	Instructor describes today's tasks and deliverables
1:10 pm – 2pm	Project work time
2pm – 4pm	Teams present what they have achieved this week

Description

Today you will continue the work started yesterday - setting up Jenkins and creating a CI pipeline. In addition you will add a stage to the pipeline to run unit tests. If time permits you should work on implementing additional API components that allow for the managing of events.

Tasks

- Continue CI Pipeline tasks from yesterday.
- Add a testing stage to the pipeline.
- Demonstrate a working application using pipeline-built components

Bonus (as time permits)

- Implement additional APIs to manage events and allow customers to register for events.
- Integrated them with the existing customer and security APIs

Deliverables

By the end of the day your pair should have a working CI pipeline with various stages including testing.

- A functional CI pipeline (pipeline build succeeds, pipeline deploys to "production" Kubernetes cluster)
- Docker images built using the pipeline
- A functional end-to-end application setup using pipeline-built components.