Tecnológico de Monterrey

Sprint 5 - Deployment & Closure
Deployment/Maintenance Guide & Documentation

Andrés Daniel Martínez Bermúdez - A00227463
Jacob García Rodríguez - A01643891
Jean Paul López Pándura - A01637266
David Sanchez Baez - A01798202
Luis Marco Barriga Baez - A01643954
Victor Javier Quintana Cisneros - A01643020

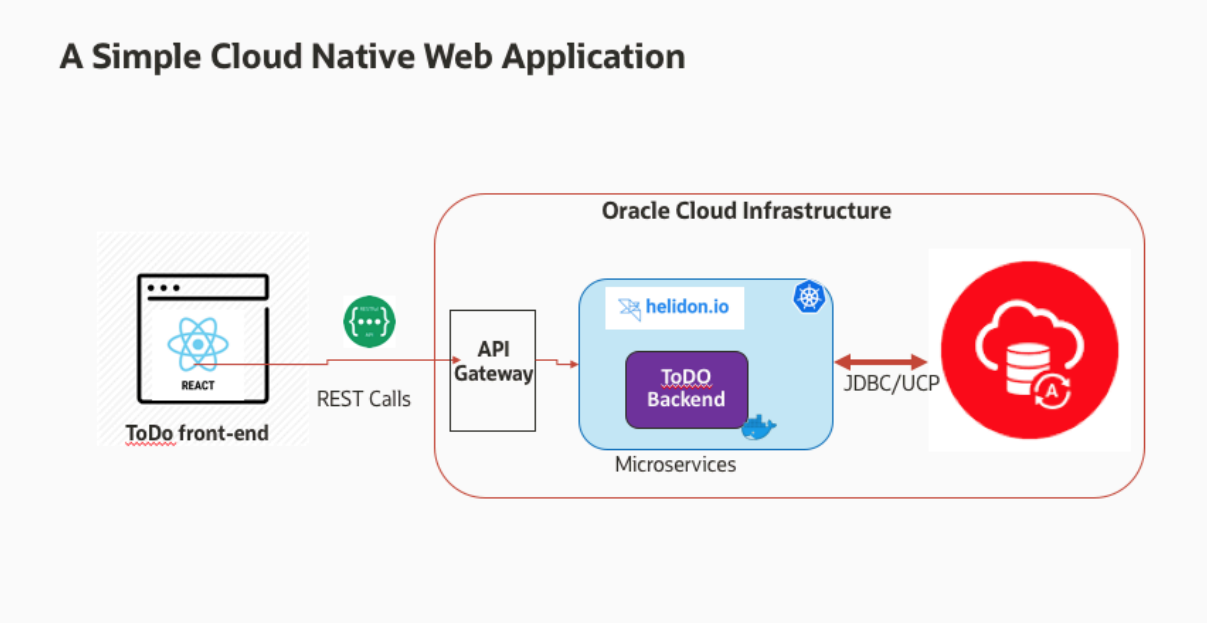
Introduction

This document is a guide to set up, deploy & maintain the Oracle Task Management Suite, an application to assist teams using AGILE or AGILE adjacent workflows with tools to track tasks and performance across sprints.

Architecture

The Suite consists of a Spring Boot application written in Java, which serves a React webpage written with Typescript for project managers. It also runs a Telegram Bot, the main interface for developers to manage their tasks.

As a Cloud Native application, the entirety of its services are dependencies are hosted in Oracle's Cloud Infrastructure, or OCI.



The application is also configured and optimized for a Continuous Integration/Continuous Deployment pipeline, hosted on OCI. On a push to the `main` branch of the GitHub repository, OCI will begin the compilation process, which outputs a Docker image, then published to the cluster. If any step during the compilation or deployment process fails then the cluster is rolled back to the last successful deployment.

Setup Guide

Requirements

Before following the rest of this guide, it is important to obtain an Oracle Cloud tenancy with enough credits. A Free Tier account does not have access to some of the features used by the application. As such, it is necessary to have a Paid account or an account with a Trial.

OCI provisioning

The first step is to provision the resources needed for the application. This is done with a collection of scripts (see Appendix A) executed on the tenancy's Cloud Shell that use Terraform to reliably and consistently provision and configure said resources.

See this Oracle Workshop for instructions on the initial provisioning of resources. While it uses a different repository, the infrastructure is exactly the same. Follow the instructions in Lab 1 but no further.

During the setup a database root password, as well as a password for the application user (by default `TODOUSER`). It is important to save these passwords as they will be used for the application and for administration.

Database setup

The next step is to setup the structure of the database. In the project's root folder is a `setup_db.sql` file, which contains the necessary commands to provision the tables that will be used by the application.

Dependencies for initial compilation

The initial deployment of the application will be done via the Tenancy's Cloud Shell. It is imperative to install **NodeJS** along with its package manager, **npm**.

Initial compilation & deployment

For simplicity, there is an alternate CI/CD pipeline setup on GitHub Actions. On every commit to `main`, the project will be compiled, tested & published to the GitHub Container Registry, or GHCR. It is important to note that the container image has the repository owner (in this case the `oracle35` organization) hard-coded, so forks will not work. Instead the organization should be transferred to the new owners.

Optional: SSL certificate

If SSL is desired then the Kubernetes cluster must be provided with an SSL certificate and key provided by a Certificate Authority (CA). This in turn means having access to a domain and its DNS entries. Instructions for provisioning the certificate are heavily dependent on the Domain Provider used and are out of scope for this guide.

Once the certificate and key are obtained, they must be uploaded to the Cloud Shell, and the following command must be run to share them to the cluster:

```
kubectl create secret tls ssl-certificate-secret --key tls.key --cert tls.crt
```

IMPORTANT: if SSL is **not** desired, in the step where the Kubernetes manifest is applied the file `todolistapp-springboot-no-ssl.yaml` **must** be used.

Static IP

The next step is to provision a static IP via OCI. The instructions as provided by Oracle are in this article . Once this is done, the Kubernetes manifest must be modified to reflect this IP. On either `todolistapp-springboot.yaml` or `todolistapp-springboot-no-ssl.yaml`, under the `MtdrSpring/backend` folder, edit the key `spec.loadBalancerIP`to reflect the reserved IP address.

The next step is to deploy the image. On the Cloud Shell, with the current directory set to the project's root directory, run the following commands, making sure to use the proper manifest depending on if SSL is used or not:

```
git pull
kubectl -n mtdrworkshop apply -f MtdrSpring/backend/todolistapp-springboot.yaml
```

This will sync the repository with the remote ref at GitHub, and update the deployment with the new image. Its tag is simply the commit's SHA hash.

If everything went correctly, the application should be visible in the reserved IP specified in the manifest.

Setting up the Pipeline

To keep this document short & relevant, a background in and previous experience with Oracle Cloud is assumed.

Preparation

Firstly, a container must be created for all CI/CD resources. Not only does it help keeping the tenancy organized, it allows for better security.

Once the container is created a dynamic group must be created to identify all CI/CD and related resources:

```
All {
  resource.compartment.id = <projectCompartmentOCID>',
  Any {
    resource.type ='devopsdeploypipeline',
    resource.type = 'devopsbuildpipeline',
    resource.type = 'devopsrepository',
    resource.type ='devopsconnection',
    resource.type = 'devopstrigger'
  }
}
```

Next, the policies that allow these resources access to the necessary information in the tenancy must be specified:

```
Allow dynamic-group dg to manage devops-family in tenancy
Allow dynamic-group dg to manage repos in tenancy
Allow dynamic-group dg to read secret-family in tenancy
Allow dynamic-group dg to manage devops-family in tenancy
Allow dynamic-group dg to manage generic-artifacts in tenancy
Allow dynamic-group dg to use ons-topics in tenancy
Allow dynamic-group dg to read secret-family in tenancy
Allow dynamic-group dg to manage all-resources in tenancy
Allow dynamic-group dg to read secret-family in tenancy
Allow dynamic-group dg to use devops-connection in tenancy
Allow dynamic-group dg to manage cluster in tenancy
Allow dynamic-group dg to read all-artifacts in tenancy
```

The next step is to obtain a GitHub Personal Access Token, to connect the repository to Oracle's platform. Once the token is obtained, it must be added to a Vault in OCI (which, again, must be created inside the CI/CD container).

DevOps Project

Due to time limitations, the full process for provisioning the pipeline cannot be specified in this guide. However, the process isn't unique to this project and as such any general tutorial or guide on this process should be enough to configure it. There are only some project specific changes to keep in mind:

- The `build_spec.yaml` specifies an output artifact called `todoapp`. When the pipeline is created, in the Managed Build step an artifact with the same name should be specified.
- In the DevOps project two artifacts must be created: one for the Docker image and another for the Kubernetes manifest. Its contents should be the following:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: todolistapp-springboot-deployment
```

```
spec:
  template:
    spec:
      containers:
        - name: todolistapp-springboot
          image: "<tenancy-region>.ocir.io/<tenancy-namespace>/todoapp/todoapp:
${GIT_HASH}"
```

- The Docker image artifact should also contain the same URL as specified in the Kubernetes manifest artifact, replacing the variables with the relevant values.

Maintenance

Current testing is limited, however, the infrastructure for implementing new tests is already in place, which will reduce the time required to implement full test suites for all elements. The Telegram Bot has a testing framework, which enables full end-to-end testing of conversations, simulating database access. Examples of existing tests are in `MtdrSpring/backend/src/test/java/com/springboot/MyTodoList/test/bot/NewTaskTest.java`.

Logs can be inspected by using `kubectl logs`. The pod ID's can be obtained by running `kubectl get pods -n mtdrworkshop`.

Currently, there are style tests for the frontend & backend, as well as unit tests. They a are run as part of the build process, both in the OCI pipeline and in GitHub Actions.

Nix

An important aspect of the application is its dependence on Nix. Some of the details are in the `docs/` folder of the project. In summary, our project chooses a unique solution for package and dependency management in Nix. It unifies all package management from any language into a single unified platform, which additionally ensures all dependencies are tracked and immutable. What this means is that if this project (assuming a clean clone of the repository and hash integrity) is compiled on any two different machines, Nix will ensure the output is entirely the same, byte for byte. It does this by treating every dependency as a "derivation", a deterministic function, which given the same inputs (dependencies, such as a compiler, libraries, etc.) will give the same output.

What this means for the project is that dependencies which are downloaded by external programs, such as Maven for Java or NPM for Node, must be "locked," a hash must be specified so that the result of the derivation is known in advance. If the hash is incorrect or missing, compilation will fail, showing the expected hash and the specified hash. Here's an example error message:

```
> ERROR: npmDepsHash is out of date
>
> The package-lock.json in src is not the same as the in /nix/store/
rsy7wkxzqpfpgdw89wwpnv6prrxm62l7-todolistapp-frontend-0.1.0-npm-deps.
>
> To fix the issue:
> 1. Use `lib.fakeHash` as the npmDepsHash value
> 2. Build the derivation and wait for it to fail with a hash mismatch
> 3. Copy the 'got: sha256-' value back into the npmDepsHash field
```

While there is a manual fix, as specified in this message, there's a better approach. Depending on if the fail was on the frontend or backend build, either of these commands must be run. Nix must be installed, with the flakes and nix command feature enabled, to update the hashes.

For the backend (Maven):

```
nix run nixpkgs#nix-update -- todoapp --version=skip --flake
```

For the frontend (NPM):

```
nix run nixpkgs#nix-update -- todoapp-frontend --version=skip --flake
```

This command will run the `nix-update` package which evaluates the packages, searches for the mismatched hash, and updates it.

Appendix A: Build & Deployment Scripts

Note: all links are permalinks to the latest commit on the master branch of the project as of **June 13th, 2025**.

Script Name	Description	Permalink
`bin/build.sh`	Development script that facilitates the loop of compiling, verifying changes, and making new edits. It configures the application for local development, adding in environment variables specified in a secret `.env` file.	link
`env.sh`	Sets up the Cloud Shell with environment variables needed by the rest of the scripts, and adds some utility functions to the PATH.	link
`setup.sh`	The main entrypoint for the initial setup of the project. Among other things, it sets up a connection to OCI, provisions the resources needed via Terraform, and configures access to the created Kubernetes cluster.	link
`destroy.sh`	Uses Terraform to take down all of the resources provisioned in `setup.sh`. Useful for starting from scratch in case of a mistake or error in the setup process.	link
`nix-run.sh`	In local runs, if the project is executed using Nix, this script will be the entry point of the application. It sets up environment variables and picks up secrets in the `.env` file, which is not tracked by Git.	link
`build_spec.yaml`	While not explicitly a script, it contains shell instructions within that are executed by a build runner in the build pipeline. This file instructs the runner to install Nix, compile the project using it, and loading the resulting Docker image, to later use it in the deployment pipeline.	link