Day 2 Bazel

Inputs, Outputs, and Actions:

Inputs: These are the files or data required by a rule to generate its outputs. They represent the dependencies of a rule.

Outputs: These are the files or data generated by a rule. They represent the artifacts produced by a build rule.

Actions: These are the commands executed by Bazel to generate outputs from inputs. They define how to transform inputs into outputs.

```
# Example rule in a BUILD file
py_binary(
   name = "hello_world",
   srcs = ["hello_world.py"],
)
```

Sandboxing for Artifact Generation:

Bazel uses sandboxing to ensure that builds are hermetic and reproducible.

Each build action runs in a controlled environment, isolated from the rest of the system to prevent unintended dependencies.

Dependencies between Artifacts:

Bazel allows specifying dependencies between different build targets.

When one target depends on another, Bazel ensures that the dependencies are built before the dependent target.

Defining External Dependencies:

External dependencies are libraries or resources that are not part of the current Bazel workspace.

They are specified in the WORKSPACE file using rules like http_archive or git_repository.

```
# Example WORKSPACE file
http_archive(
   name = "com_github_example_library",
   urls = ["https://github.com/example/library/archive/v1.0.0.tar.gz"],
   sha256 = "abcdef123456...",
)
```

Find Dependencies between Artifacts:

Bazel provides commands like bazel query to find dependencies between different build targets.

This is useful for understanding the build graph and troubleshooting dependency issues.

Basic Commands:

Bazel provides several basic commands for building, running, and testing code, such as bazel build, bazel run, and bazel test.

These commands are used to execute build actions, run binaries, and execute tests respectively.

How to Reference Bazel Targets:

Bazel targets can be referenced using their label, which consists of the target's name and its location in the workspace.

Labels are used in BUILD files to specify dependencies between targets.

Hands on Files First Example

1. Java-tutorial/src/main/java/com/example/salutations.java

```java
package main.java.com.example;
import org.beryx.textio.TextIO;
import org.beryx.textio.TextIoFactory;

public class Salutations {

    public static void main(String[] args) {
        TextIO textIO = TextIoFactory.getTextIO();

        String user = textIO.newStringInputReader()
            .withDefaultValue("admin")
            .read("Username");


        String password = textIO.newStringInputReader()
            .withMinLength(6)
            .withInputMasking(true)
            .read("Password");
        System.out.println("Hello, " + user + "! Your password is: " + password);
    }
}
```

2. Java-tutorial /MODULE.bazel file

```
bazel_dep(name = "rules_jvm_external", version = "5.3")
maven = use_extension("@rules_jvm_external//:extensions.bzl", "maven")
maven.install(
    artifacts = [
        "org.beryx:text-io:3.4.1",
        "org.apache.commons:commons-lang3:3.12.0"  ,



    ],
    fetch_sources = True,
    repositories = [
       "https://maven.google.com",
        "https://repo1.maven.org/maven2",
        "https://jcenter.bintray.com/",
        "http://uk.maven.org/maven2",


    ],
)
use_repo(maven, "maven")
```

3. java-tutorial/BUILD

Add these to BUILD file

```
java_binary(
    name="salutations",
    main_class = "main.java.com.example.Salutations",
    runtime_deps = [":textio"],
    visibility = ["//visibility:public"],
)

java_library(
    name="textio",
    srcs = ["src/main/java/com/example/Salutations.java"],
    deps = [":greeter","@maven//:org_beryx_text_io"],
    #visibility = ["//src/main/java/com/example/cmdline:__pkg__"],
)
```

3. Run the target salutations from java-tutotrial

$ bazel run //:salutations

Hands on Files second Example

1. **Java-tutorial/src/main/java/com/example/StringUtilsExample.java**

```java
package main.java.com.example;

import org.apache.commons.lang3.StringUtils;

public class StringUtilsExample {
    public static void main(String[] args) {
        // Example strings
        String string = "The quick brown fox jumps over the lazy dog";
        String emptyString = "";
        String blankString = " ";
        String[] array = {"apple", "banana", "orange"};

        // Count occurrences of character 'o'
        int charNum = StringUtils.countMatches(string, 'o');
        System.out.println("Occurrences of 'o': " + charNum); // Output: Occurrences of 'o': 4

        // Count occurrences of substring "the"
        int stringNum = StringUtils.countMatches(string.toLowerCase(), "the");
        System.out.println("Occurrences of 'the': " + stringNum); // Output: Occurrences of 'the': 2

        // Check if strings are empty or blank
        System.out.println(StringUtils.isEmpty(emptyString)); // Output: true
        System.out.println(StringUtils.isEmpty(blankString)); // Output: false
        System.out.println(StringUtils.isBlank(emptyString)); // Output: true
        System.out.println(StringUtils.isBlank(blankString)); // Output: true

        // Trim leading and trailing whitespaces
        String trimmedString = StringUtils.trim("  Hello  ");
        System.out.println("Trimmed string: " + trimmedString); // Output: Trimmed string: Hello

        // Join array elements into a single string
        String joinedArray = StringUtils.join(array, ", ");
        System.out.println("Joined array: " + joinedArray); // Output: Joined array: apple, banana, orange
    }
}

```

2. java-tutorial/BUILD

Add this to BUILD

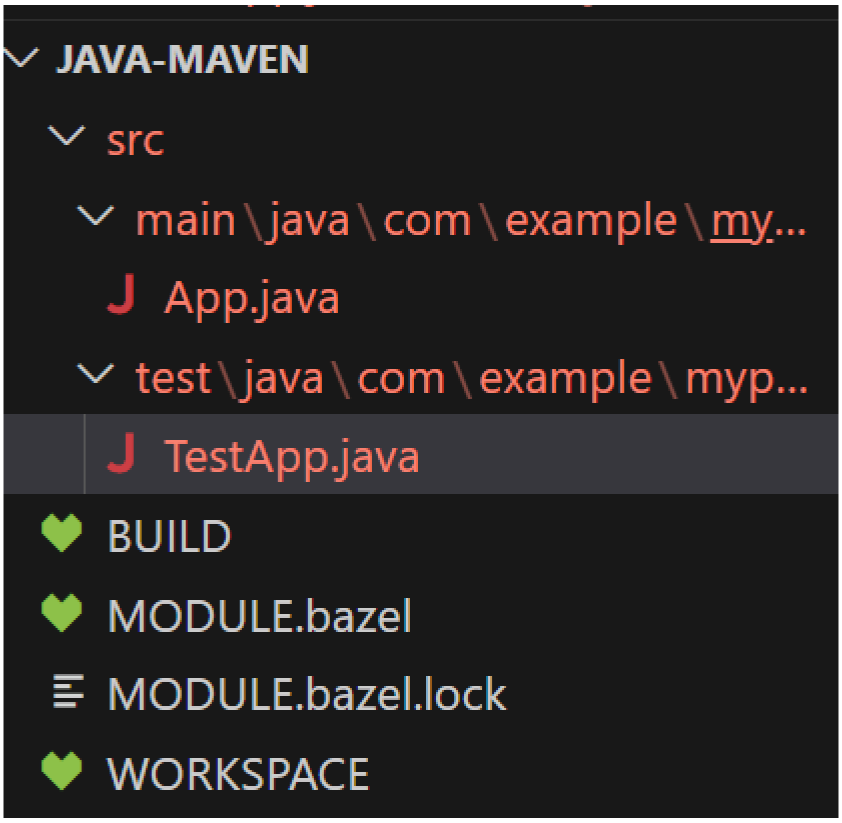
```
java_binary(
    name="stringutilsexample",
    main_class = "main.java.com.example.StringUtilsExample",
    runtime_deps = [":stringutils"],
    visibility = ["//visibility:public"],
)
java_library(
    name="stringutils",
    srcs=["src/main/java/com/example/StringUtilsExample.java"],
    deps=["@maven//:org_apache_commons_commons_lang3"],

)
```

3. run the target

$bazel run //:stringutilsexample

Hands on Files third Example

1. Create a java-maven folder
2. Create WORKSAPCE and MODULE.bazel file



```
JAVA-MAVEN
  src
    main\java\com\example\my...
      App.java
    test\java\com\example\myp...
      TestApp.java
  BUILD
  MODULE.bazel
  MODULE.bazel.lock
  WORKSPACE
```

MODULE.bazel file

```
bazel_dep(name = "rules_jvm_external", version = "5.3")

maven = use_extension("@rules_jvm_external//:extensions.bzl", "maven")
maven.install(
    artifacts = [
        "junit:junit:4.12",
        "com.google.guava:guava:28.0-jre",
    ],
    fetch_sources = True,
    repositories = [
        "https://maven.google.com",
        "https://repo1.maven.org/maven2",
        "https://jcenter.bintray.com/",
        "http://uk.maven.org/maven2",
    ],
)
use_repo(maven, "maven")
```

```
```

BUILD file

```
load("@rules_java//java:defs.bzl", "java_binary", "java_library", "java_test")


package(default_visibility = ["//visibility:public"])

java_library(
    name = "java-maven-lib",
    srcs = glob(["src/main/java/com/example/myproject/*.java"]),
    deps = ["@maven//:com_google_guava_guava",
            "@maven//:junit_junit"
        ],
)

java_binary(
    name = "java-maven",
    main_class = "main.java.com.example.myproject.App",
    runtime_deps = [":java-maven-lib"],
)

java_test(
    name = "tests",
    srcs = glob(["src/test/java/com/example/myproject/*.java"]),
    test_class = "test.java.com.example.myproject.TestApp",
    deps = [":java-maven-lib"],
)
```

App.java

```java
package main.java.com.example.myproject;


import com.google.common.primitives.Ints;
```

```
/**
 * This application compares two numbers, using the Ints.compare
 * method from Guava.
 */
public class App {

  public static int compare(int a, int b) {
    return Ints.compare(a, b);
  }

  public static void main(String... args) throws Exception {
    App app = new App();
    System.out.println("Success: " + app.compare(2, 1));
  }

}
```

TestApp.java

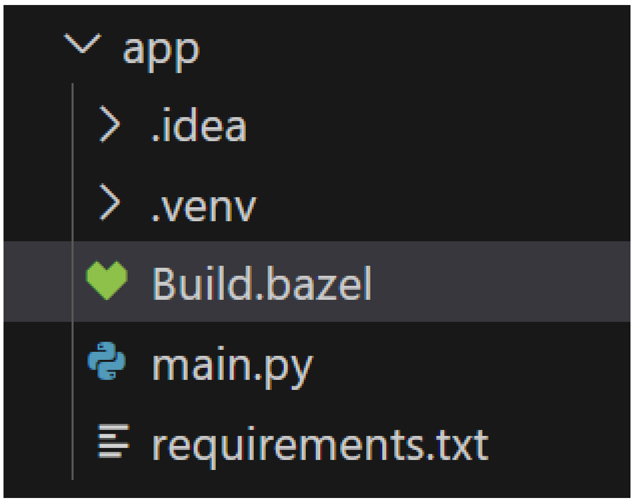
```
package test.java.com.example.myproject;
import main.java.com.example.myproject.App;
import static org.junit.Assert.assertEquals;
import org.junit.Test;

/**
 * Tests for correct dependency retrieval with maven rules.
 */
public class TestApp {

  @Test
  public void testCompare() throws Exception {
    App app = new App();
    assertEquals("should return 0 when both numbers are equal", 0,
app.compare(1, 1));
  }
}
```

Run the target tests

$bazel test //:tests

Hands on Files fourth Example

1. **Open pythontutorial folder and create app folder**



```
app
  .idea
  .venv
  Build.bazel
  main.py
  requirements.txt
```

2. **app/BUILD**

```
load("@my_deps//:requirements.bzl", "requirement")
load("@rules_python//python:defs.bzl", "py_binary", "py_library")

py_binary(
  name = "main",
  srcs = ["main.py"],
  deps =  [ "//calc:calculator",
            requirement("Flask"),
        ],

)
```

3. Pythontutorial/WORKSPACE

```

load("@bazel_tools//tools/build_defs/repo:http.bzl", "http_archive")

http_archive(
    name = "rules_python",
    sha256 = "c68bdc4fbec25de5b5493b8819cfc877c4ea299c0dcb15c244c5a00208cde311",
    strip_prefix = "rules_python-0.31.0",
    url = "https://github.com/bazelbuild/rules_python/releases/download/0.31.0/rules_python-0.31.0.tar.gz",
)

load("@rules_python//python:repositories.bzl", "py_repositories")

py_repositories()

load("@rules_python//python:pip.bzl", "pip_parse")
pip_parse(
   name = "my_deps",
   requirements_lock = "//app:requirements.txt",
)

load("@my_deps//:requirements.bzl", "install_deps")
install_deps()

```

4. main.py

```python
from calc.calculator import Calculator
from flask import Flask
from random import randint

app = Flask(__name__)
calculator = Calculator()

@app.route('/')
def randomNumberCalculator():
  randomInt1 = randint(0, 250)
  randomInt2 = randint(0, 250)
  return "{} + {} = {}?".format(randomInt1, randomInt2, \
  calculator.add(randomInt1, randomInt2))

if __name__ == '__main__':
  app.run(host='0.0.0.0')
```

5. requirements.txt

```
blinker==1.7.0
certifi==2024.2.2
charset-normalizer==2.0.12
click==8.1.7
colorama==0.4.6
Flask==3.0.3
idna==3.7
itsdangerous==2.2.0
Jinja2==3.1.3
MarkupSafe==2.1.5
```

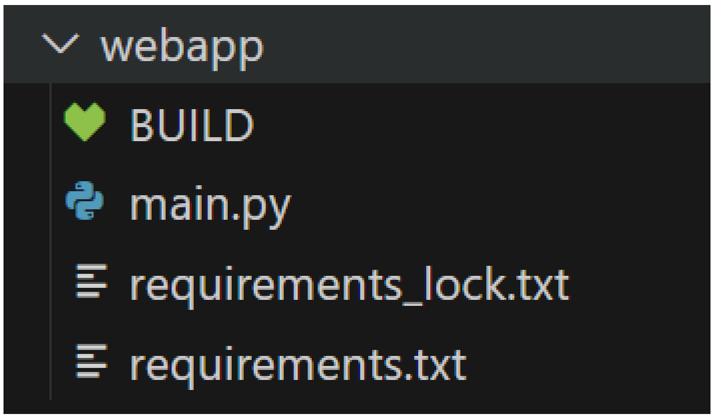
```
requests==2.26.0
urllib3==1.26.18
Werkzeug==3.0.2
```

6. run the target main

$ bazel run //app:main

Hands on Files fifth Example

1. **Open pythontutorial folder and create webapp folder**



```
webapp
    BUILD
    main.py
    requirements_lock.txt
    requirements.txt
```

2. **webapp/BUILD**

```
load("@rules_python//python:defs.bzl", "py_library", "py_binary")
load("@my_deps//:requirements.bzl", "requirement")

py_binary(
    name = "main",
    srcs = ["main.py"],
    deps = [requirement("fastapi"), requirement("uvicorn")],
)
```

3. requirements.txt

```
fastapi==0.109.0
accelerate==0.27.2
altair==5.2.0
Appium-Python-Client==3.0.0
attrs==23.1.0
blinker==1.7.0
cachetools==5.3.2
certifi==2023.5.7
cffi==1.15.1
charset-normalizer==3.3.2
click==8.1.7
colorama==0.4.6
distlib==0.3.6
et-xmlfile==1.1.0
exceptiongroup==1.1.3
filelock==3.12.2
Flask==3.0.3
fsspec==2024.2.0
gitdb==4.0.11
GitPython==3.1.42
h11==0.14.0
huggingface-hub==0.20.3
idna==3.4
importlib-metadata==7.0.1
itsdangerous==2.2.0
Jinja2==3.1.3
jsonschema==4.21.1
jsonschema-specifications==2023.12.1
markdown-it-py==3.0.0
MarkupSafe==2.1.5
mdurl==0.1.2
mpmath==1.3.0
```

```
networkx==3.2.1
numpy==1.26.1
openpyxl==3.1.2
outcome==1.2.0
packaging==23.2
pandas==2.1.1
pillow==10.2.0
pipenv==2023.7.11
platformdirs==3.8.1
protobuf==4.25.3
psutil==5.9.8
pyarrow==15.0.0
pycparser==2.21
pydeck==0.8.1b0
Pygments==2.17.2
PySocks==1.7.1
python-dateutil==2.8.2
pytz==2023.3.post1
PyYAML==6.0.1
referencing==0.33.0
regex==2023.12.25
requests==2.31.0
rich==13.7.0
robotframework==6.1.1
robotframework-pythonlibcore==4.2.0
robotframework-seleniumlibrary==6.1.2
rpds-py==0.18.0
safetensors==0.4.2
selenium==4.12.0
six==1.16.0
smmap==5.0.1
sniffio==1.3.0
sortedcontainers==2.4.0
streamlit==1.31.1
sympy==1.12
tenacity==8.2.3
tokenizers==0.15.2
toml==0.10.2
```

```
toolz==0.12.1
torch==2.2.1
tornado==6.4
tqdm==4.66.2
transformers==4.38.1
trio==0.22.2
trio-websocket==0.10.3
typing_extensions==4.9.0
tzdata==2023.3
tzlocal==5.2
urllib3==2.0.4
pydantic_core==2.18.1
validators==0.22.0
virtualenv==20.23.1
virtualenv-clone==0.5.7
uvicorn==0.29.0
watchdog==4.0.0
Werkzeug==3.0.2
wsproto==1.2.0
zipp==3.17.0
pydantic==2.7.0
starlette==0.37.2
anyio==4.3.0
annotated_types==0.6.0
```

4. webapp/main.py

```python
from fastapi import FastAPI
import uvicorn
app = FastAPI()


#@app.get("/status")
#def read_root():
    #return {"status": "UP", "version": "0.1.0"}

@app.get("/")
async def root():
    return {"message": "Hello, World!"}

if __name__ == "__main__":
    uvicorn.run(app, host="127.0.0.1", port=8000)
```

5. modify pythontutorial/WORKSPACE file

```
load("@bazel_tools//tools/build_defs/repo:http.bzl", "http_archive")

http_archive(
    name = "rules_python",
    sha256 =
"c68bdc4fbec25de5b5493b8819cfc877c4ea299c0dcb15c244c5a00208cde311",
    strip_prefix = "rules_python-0.31.0",
    url =
"https://github.com/bazelbuild/rules_python/releases/download/0.31.0/rules_pyt
hon-0.31.0.tar.gz",
)

load("@rules_python//python:repositories.bzl", "py_repositories")

py_repositories()

load("@rules_python//python:pip.bzl", "pip_parse")
pip_parse(
   name = "my_deps",
   requirements_lock = "//webapp:requirements.txt",
)

load("@my_deps//:requirements.bzl", "install_deps")
install_deps()
```

6. run the target main

```
$ bazel run //webapp:main
```

Hands on Files sixth Example (query)

```
bazelqueryguide
├── BUILD
├── src
│   └── main
│       └── java
│           └── com
│               └── example
│                   ├── customers
│                   │   ├── Jenny.java
│                   │   ├── Amir.java
│                   │   └── BUILD
│                   ├── dishes
│                   │   ├── Pizza.java
│                   │   ├── MacAndCheese.java
│                   │   └── BUILD
│                   ├── ingredients
│                   │   ├── Cheese.java
│                   │   ├── Tomatoes.java
│                   │   ├── Dough.java
│                   │   ├── Macaroni.java
│                   │   └── BUILD
│                   ├── restaurant
│                   │   ├── Cafe.java
│                   │   ├── Chef.java
│                   │   └── BUILD
│                   ├── reviews
│                   │   ├── Review.java
│                   │   └── BUILD
│                   └── Runner.java
└── MODULE.bazel
```

https://github.com/bazelbuild/examples/tree/main/query-quickstart

dependencies:

1. bazel run //:runner
2. bazel query //src/main/java/com/example/dishes/...
3. bazel query //src/main/java/com/example/ingredients/...

4. bazel query --noimplicit_deps "deps(:runner)"

reverse dependencies

bazel query "rdeps(universe_scope, target)"

example: bazel query "rdeps(//... ,
//src/main/java/com/example/ingredients:cheese)"

tags

bazel query 'attr(tags, "pizza",
//src/main/java/com/example/customers/...)'

Q- 1 Find out what jenny wants to order

Add a new dependency

Smoothie—made up of banana and strawberry

src/main/java/com/example/ingredients/Strawberry.java

```
package main.java.com.example.ingredients;
public class Strawberry {


}
```

src/main/java/com/example/ingredients/Banana.java

```
package main.java.com.example.ingredients;
public class Banana {
}
```

src/main/java/com/example/dishes/Smoothie.java

```
package com.example.dishes;

public class Smoothie {
    public static final String DISH_NAME = "Smoothie";
    public static final String DESCRIPTION = "Yummy and Refreshing";
}
```

src/main/java/com/example/ingredients/BUILD

```
java_library(
    name = "cheese",
    visibility = ["//visibility:public"],
    srcs = ["Cheese.java"],
)

java_library(
    name = "dough",
    visibility = ["//visibility:public"],
    srcs = ["Dough.java"],
)

java_library(
    name = "macaroni",
    visibility = ["//visibility:public"],
    srcs = ["Macaroni.java"],
)

java_library(
    name = "tomato",
    visibility = ["//visibility:public"],
    srcs = ["Tomato.java"],
)

java_library(
    name = "strawberry",
    visibility = ["//visibility:public"],
    srcs = ["Strawberry.java"],
)

java_library(
    name = "banana",
    visibility = ["//visibility:public"],
    srcs = ["Banana.java"],
)
```

src/main/java/com/example/dishes/BUILD

```
java_library(
   name = "macAndCheese",
   visibility = ["//visibility:public"],
   srcs = ["MacAndCheese.java"],
   deps = [
      "//src/main/java/com/example/ingredients:cheese",
      "//src/main/java/com/example/ingredients:macaroni",
   ],
)

java_library(
   name = "pizza",
   visibility = ["//visibility:public"],
   srcs = ["Pizza.java"],
   deps = [
      "//src/main/java/com/example/ingredients:cheese",
      "//src/main/java/com/example/ingredients:dough",
      "//src/main/java/com/example/ingredients:tomato",
   ],
)

java_library(
   name = "smoothie",
   visibility = ["//visibility:public"],
   srcs = ["Smoothie.java"],
   deps = [
      "//src/main/java/com/example/ingredients:strawberry",
      "//src/main/java/com/example/ingredients:banana",
   ],
)
```

src/main/java/com/example/restaurant/BUILD

```
java\_library(
   name = "chef",
   visibility = ["//visibility:public"],
   srcs = [
      "Chef.java",
   ],

   deps = [
      "//src/main/java/com/example/dishes:macAndCheese",
      "//src/main/java/com/example/dishes:pizza",
      "//src/main/java/com/example/dishes:smoothie",
   ],
)

java\_library(
   name = "cafe",
   visibility = ["//visibility:public"],
   srcs = [
      "Cafe.java",
   ],
   deps = [
      ":chef",
   ],
)
```

```
bazel query --noimplicit_deps 'deps(:runner)'
```

Finding the paths between dependencies

```
bazel query "somepath(//src/main/java/com/example/restaurant/...,
//src/main/java/com/example/ingredients:cheese)"

bazel query "allpaths(//src/main/java/com/example/restaurant/...,
//src/main/java/com/example/ingredients:cheese)"
```

Q-2

One of Cafe Bazel’s customers gave the restaurant's first review! Unfortunately, the review is missing some details such as the identity of the reviewer and what dish it’s referencing. Luckily, you can access this information with Bazel. The reviews package contains a program that prints a review from a mystery customer. Build and run it with:

```
bazel build //src/main/java/com/example/reviews:review

bazel-bin/src/main/java/com/example/reviews/review
```

Going off Bazel queries only, try to find out who wrote the review, and what dish they were describing.

Check the tags and dependencies for useful information.