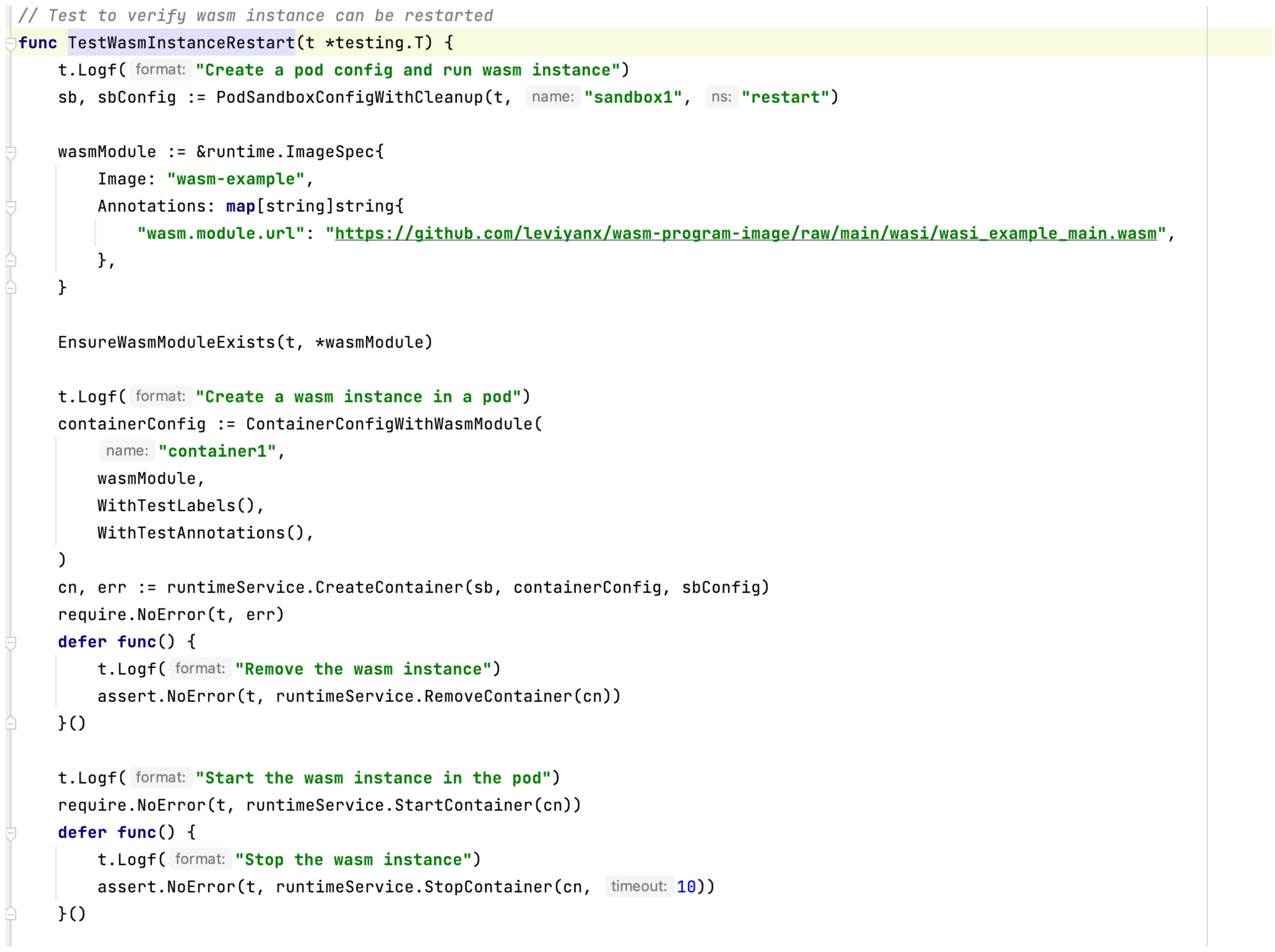
进展

- sandbox部分的代码目前不用改动（能够直接使用基于runwasi实现的shim）
- 完成了wasm instance的集成测试代码
- 基本完成**create** wasm instance部分的代码，并通过集成测试
  - 包含完善wasm instance、wasm instance store
- 验证k8s能用runwasi实现的shim来部署pod、wasm container

wasm instance 部分的集成测试

代码

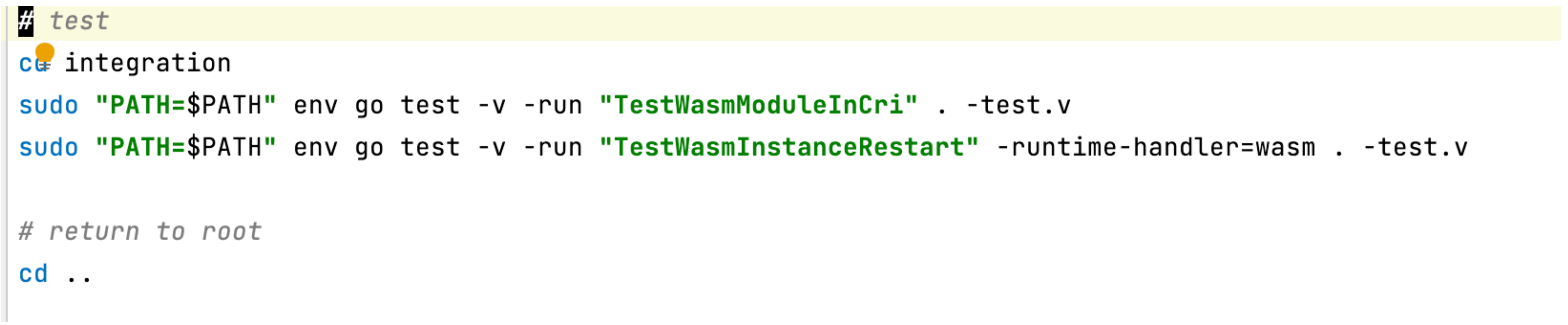
- integration/container_restart_test.go/ **TestWasmInstanceRestart**
- 包含：
  - run sandbox
  - pull wasm module
  - create wasm instance
  - remove wasm instance
  - start wasm instance
  - stop wasm instance



```go
// Test to verify wasm instance can be restarted
func TestWasmInstanceRestart(t *testing.T) {
    t.Logf( format: "Create a pod config and run wasm instance")
    sb, sbConfig := PodSandboxConfigWithCleanup(t,  name: "sandbox1",  ns: "restart")

    wasmModule := &runtime.ImageSpec{
        Image: "wasm-example",
        Annotations: map[string]string{
            "wasm.module.url": "https://github.com/leviyanx/wasm-program-image/raw/main/wasi/wasi_example_main.wasm",
        },
    }

    EnsureWasmModuleExists(t, *wasmModule)

    t.Logf( format: "Create a wasm instance in a pod")
    containerConfig := ContainerConfigWithWasmModule(
         name: "container1",
        wasmModule,
        WithTestLabels(),
        WithTestAnnotations(),
    )
    cn, err := runtimeService.CreateContainer(sb, containerConfig, sbConfig)
    require.NoError(t, err)
    defer func() {
        t.Logf( format: "Remove the wasm instance")
        assert.NoError(t, runtimeService.RemoveContainer(cn))
    }()

    t.Logf( format: "Start the wasm instance in the pod")
    require.NoError(t, runtimeService.StartContainer(cn))
    defer func() {
        t.Logf( format: "Stop the wasm instance")
        assert.NoError(t, runtimeService.StopContainer(cn,  timeout: 10))
    }()
```

wasm instance 部分的集成测试

测试脚本

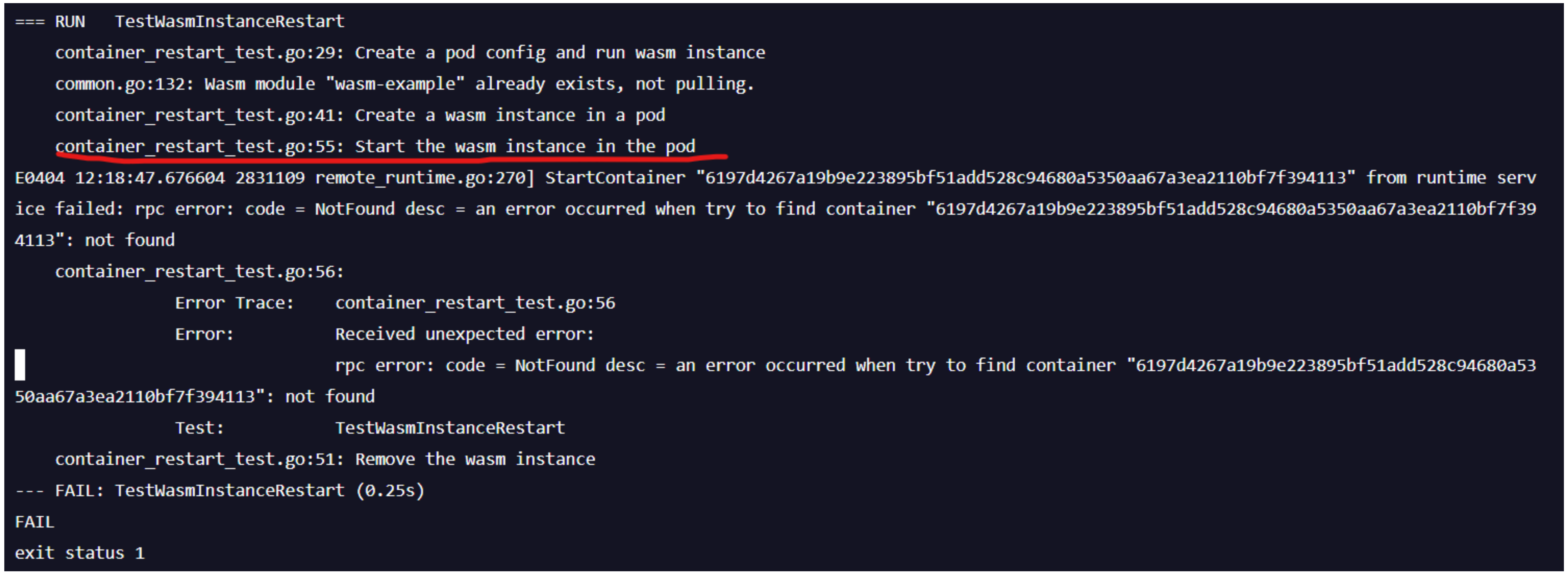
- /cri-integration-test-after-adding-wasm.sh
  - 在test wasm module的基础上，添加了wasm instance的测试
  - 

```
# test
cd integration
sudo "PATH=$PATH" env go test -v -run "TestWasmModuleInCri" . -test.v
sudo "PATH=$PATH" env go test -v -run "TestWasmInstanceRestart" -runtime-handler=wasm . -test.v

# return to root
cd ..
```

- 测试步骤：
  - 启动containerd：bash start-containerd.sh
  - 对cri测试：bash cri-integration-test-after-adding-wasm.sh
  - 关闭containerd：bash stop-containerd.sh

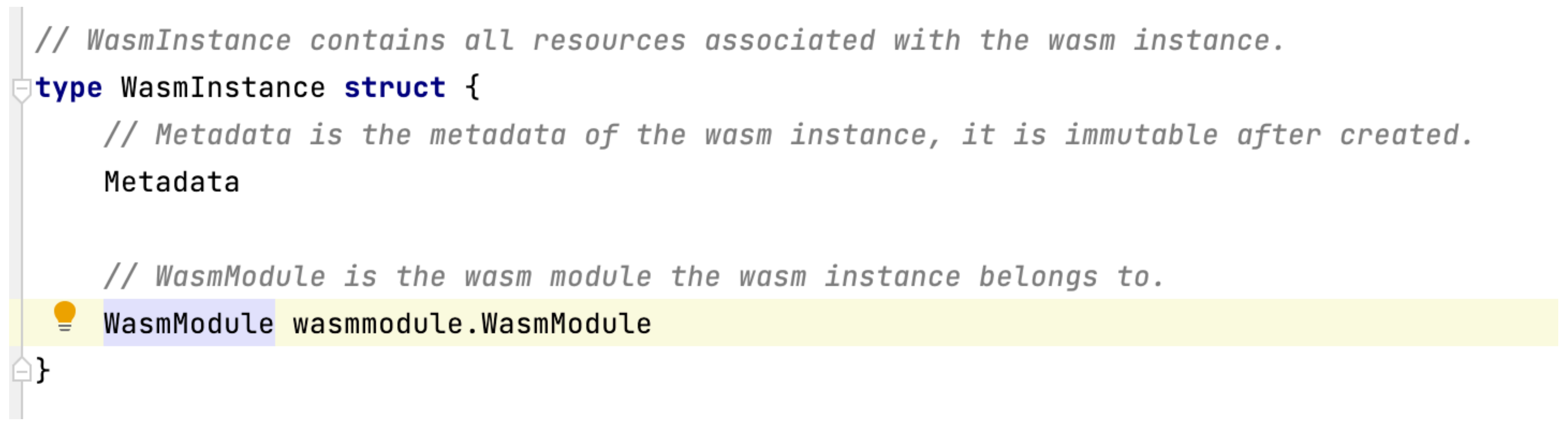
wasm instance 部分的集成测试

create wasm instance通过测试

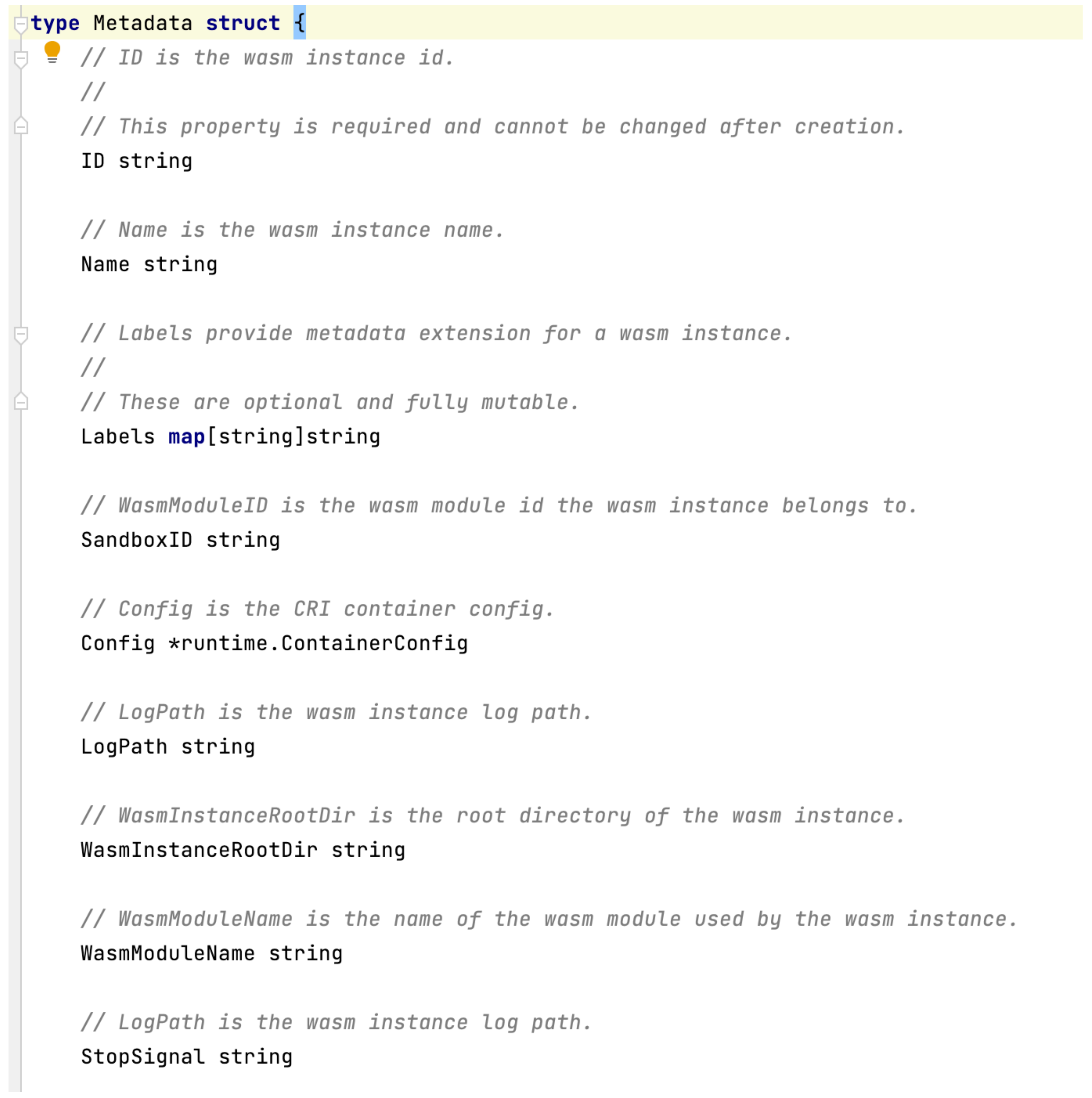


```
=== RUN   TestWasmInstanceRestart
    container_restart_test.go:29: Create a pod config and run wasm instance
    common.go:132: Wasm module "wasm-example" already exists, not pulling.
    container_restart_test.go:41: Create a wasm instance in a pod
    container_restart_test.go:55: Start the wasm instance in the pod
E0404 12:18:47.676604 2831109 remote_runtime.go:270] StartContainer "6197d4267a19b9e223895bf51add528c94680a5350aa67a3ea2110bf7f394113" from runtime serv
ice failed: rpc error: code = NotFound desc = an error occurred when try to find container "6197d4267a19b9e223895bf51add528c94680a5350aa67a3ea2110bf7f39
4113": not found
    container_restart_test.go:56:
                Error Trace:    container_restart_test.go:56
                Error:          Received unexpected error:
                                rpc error: code = NotFound desc = an error occurred when try to find container "6197d4267a19b9e223895bf51add528c94680a53
50aa67a3ea2110bf7f394113": not found
                Test:           TestWasmInstanceRestart
    container_restart_test.go:51: Remove the wasm instance
--- FAIL: TestWasmInstanceRestart (0.25s)
FAIL
exit status 1
```

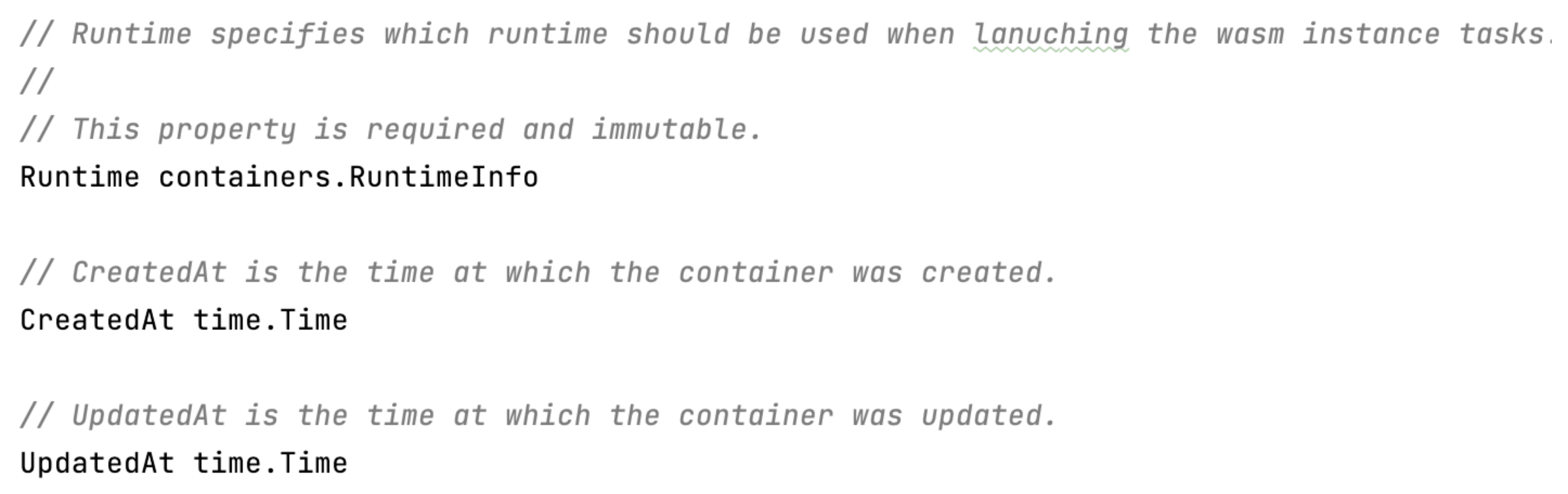
wasm instance



```go
// WasmInstance contains all resources associated with the wasm instance.
type WasmInstance struct {
    // Metadata is the metadata of the wasm instance, it is immutable after created.
    Metadata

    // WasmModule is the wasm module the wasm instance belongs to.
    WasmModule wasmmodule.WasmModule
}
```



```go
type Metadata struct {
    // ID is the wasm instance id.
    //
    // This property is required and cannot be changed after creation.
    ID string

    // Name is the wasm instance name.
    Name string

    // Labels provide metadata extension for a wasm instance.
    //
    // These are optional and fully mutable.
    Labels map[string]string

    // WasmModuleID is the wasm module id the wasm instance belongs to.
    SandboxID string

    // Config is the CRI container config.
    Config *runtime.ContainerConfig

    // LogPath is the wasm instance log path.
    LogPath string

    // WasmInstanceRootDir is the root directory of the wasm instance.
    WasmInstanceRootDir string

    // WasmModuleName is the name of the wasm module used by the wasm instance.
    WasmModuleName string

    // LogPath is the wasm instance log path.
    StopSignal string
```



```go
    // Runtime specifies which runtime should be used when lanuching the wasm instance tasks
    //
    // This property is required and immutable.
    Runtime containers.RuntimeInfo

    // CreatedAt is the time at which the container was created.
    CreatedAt time.Time

    // UpdatedAt is the time at which the container was updated.
    UpdatedAt time.Time
```

Create wasm instance的大致流程

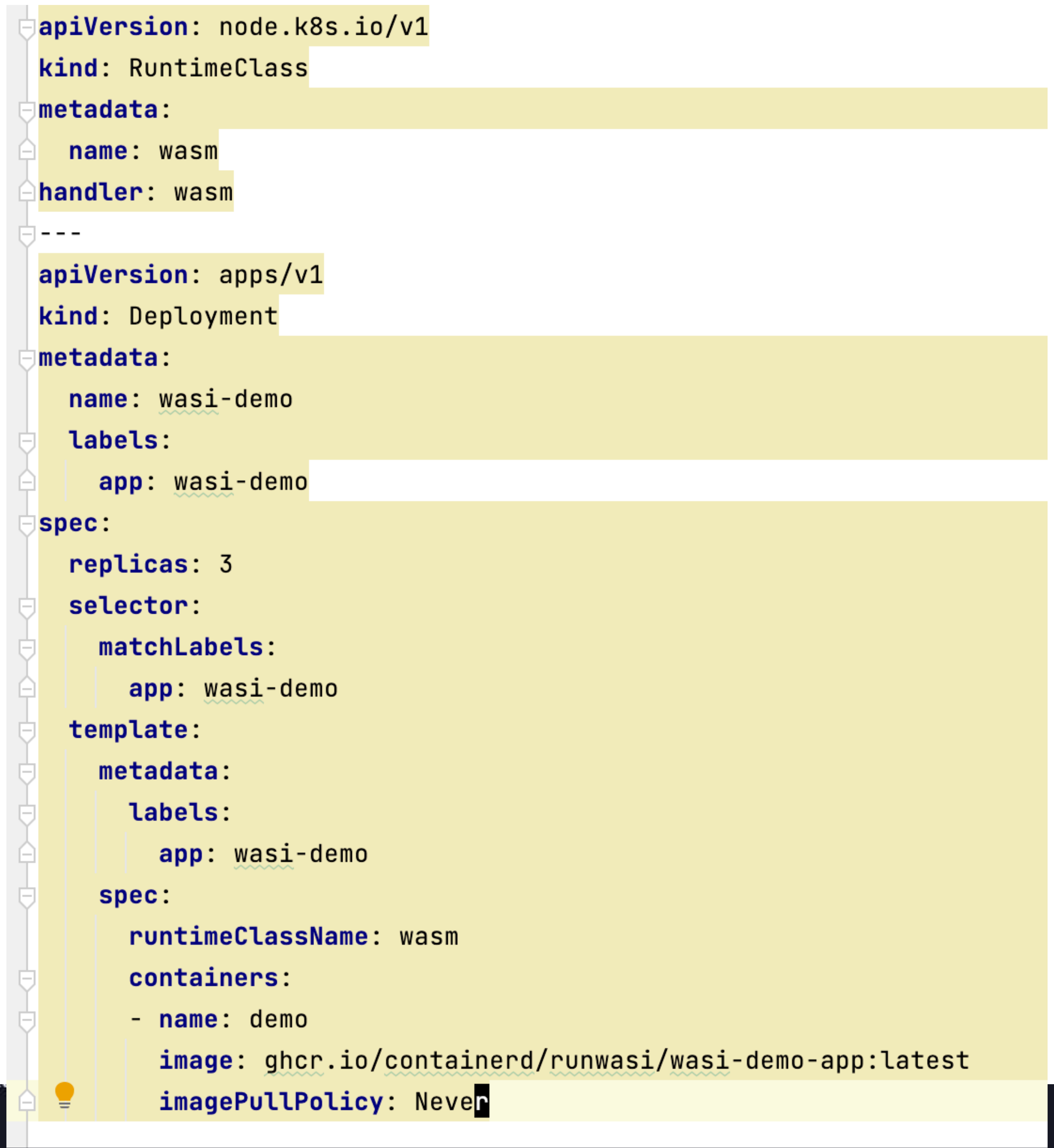
- 获取running sandbox的配置、pid
- 判断create container config中的image是否是wasm module，如果是，则创建wasm instance，否则创建oci container
- 如果是创建wasm instance
    - 生成唯一id、name，并将其存储到一个index库中，保证name唯一并且没有创建过（即保证这个wasm instance之前没有创建过），以及name和id一一对应
    - 初始化metadata
    - 获取wasm module
    - 获取wasm runtime（原来是oci runtime）其实就是shim

Create wasm instance的大致流程

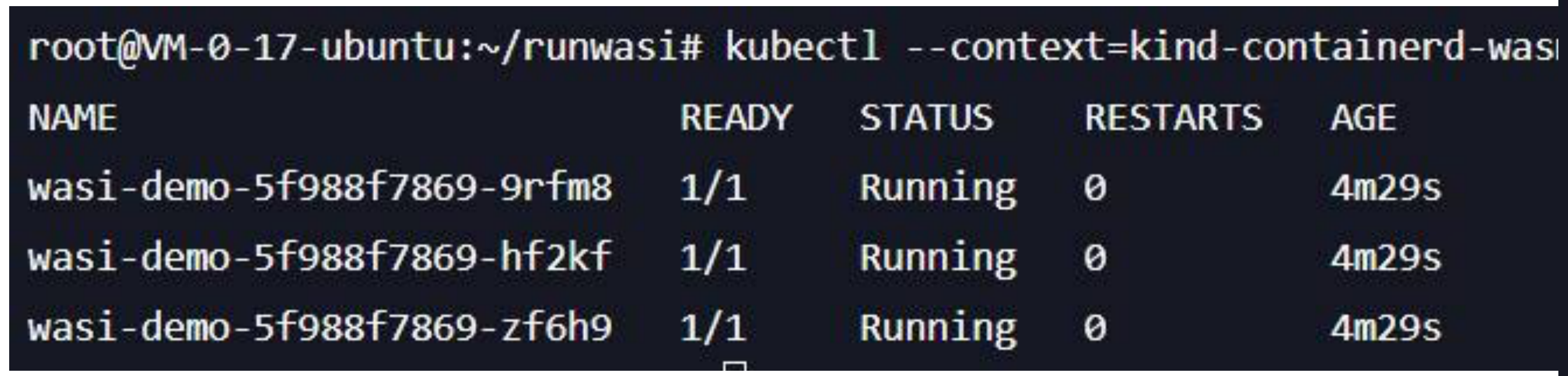
- 如果是创建wasm instance
  - 初始化wasm instance
  - 把wasm instance存储到store中
  - metric timer记录wasm instance的更新时间
  - 返回container id（wasm instance id）

验证k8s能用runwasi实现的shim来部署pod、wasm container

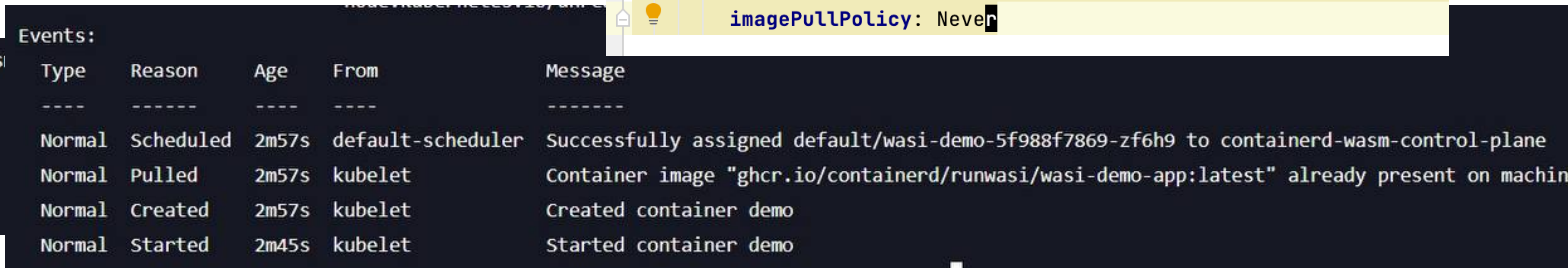
- 编译、安装shim
- 设置containerd使用wasmtime shim
  - runtime name设置为wasm
- 启动k8s，配置其使用containerd
- kubectl apply -f deploy.yaml



```yaml
apiVersion: node.k8s.io/v1
kind: RuntimeClass
metadata:
  name: wasm
handler: wasm
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wasi-demo
  labels:
    app: wasi-demo
spec:
  replicas: 3
  selector:
    matchLabels:
      app: wasi-demo
  template:
    metadata:
      labels:
        app: wasi-demo
    spec:
      runtimeClassName: wasm
      containers:
      - name: demo
        image: ghcr.io/containerd/runwasi/wasi-demo-app:latest
        imagePullPolicy: Never
```

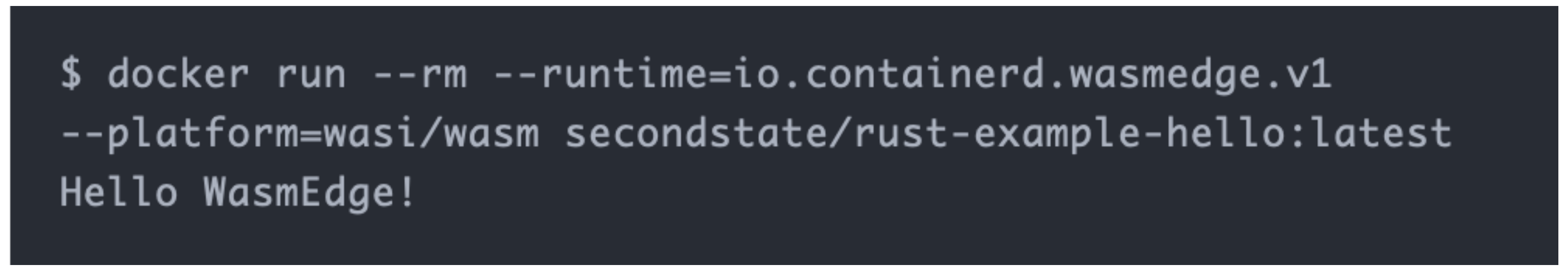


```
root@VM-0-17-ubuntu:~/runwasi# kubectl --context=kind-containerd-wasi
NAME                          READY   STATUS    RESTARTS   AGE
wasi-demo-5f988f7869-9rfm8    1/1     Running   0          4m29s
wasi-demo-5f988f7869-hf2kf    1/1     Running   0          4m29s
wasi-demo-5f988f7869-zf6h9    1/1     Running   0          4m29s
```

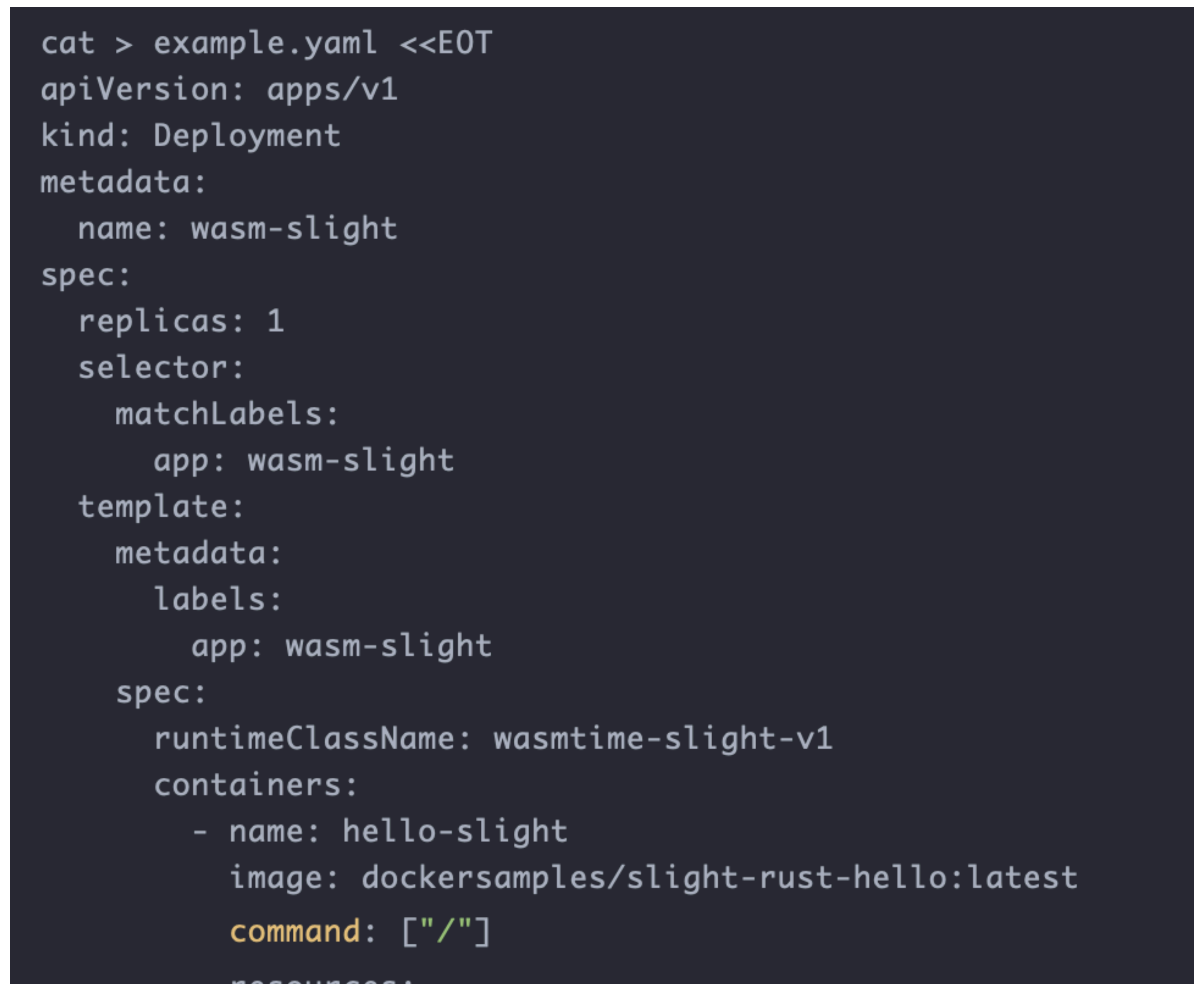


```
Events:
  Type    Reason     Age    From               Message
  ----    ------     ----   ----               -------
  Normal  Scheduled  2m57s  default-scheduler  Successfully assigned default/wasi-demo-5f988f7869-zf6h9 to containerd-wasm-control-plane
  Normal  Pulled     2m57s  kubelet            Container image "ghcr.io/containerd/runwasi/wasi-demo-app:latest" already present on machin
  Normal  Created    2m57s  kubelet            Created container demo
  Normal  Started    2m45s  kubelet            Started container demo
```

docker+wasm

- 第一第二版都是充分利用containerd和shim
- 

```
$ docker run --rm --runtime=io.containerd.wasmedge.v1
--platform=wasi/wasm secondstate/rust-example-hello:latest
Hello WasmEdge!
```

  - ctr run --rm --runtime=io... ...
- 

```
cat > example.yaml <<EOT
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wasm-slight
spec:
  replicas: 1
  selector:
    matchLabels:
      app: wasm-slight
  template:
    metadata:
      labels:
        app: wasm-slight
    spec:
      runtimeClassName: wasmtime-slight-v1
      containers:
        - name: hello-slight
          image: dockersamples/slight-rust-hello:latest
          command: ["/"]
```