

Trouble testing **Kubernetes** on your
bespoke cloud?

Kubetest2 to the rescue!

Trouble testing **Kubernetes** on your
bespoke cloud?

Kubetest2 to the rescue!

Who am I?



Priyanka Saggu
@psaggu

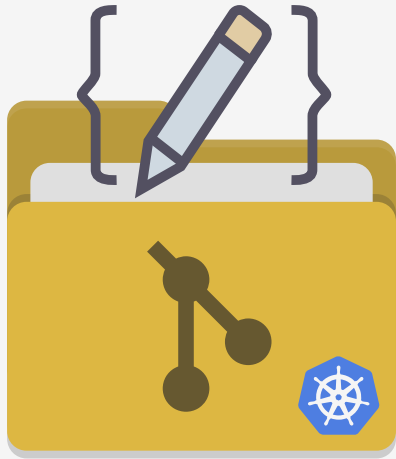
Kubernetes

E2E (end-to-end) test

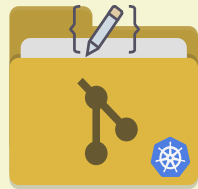
Kubernetes: E2E Test



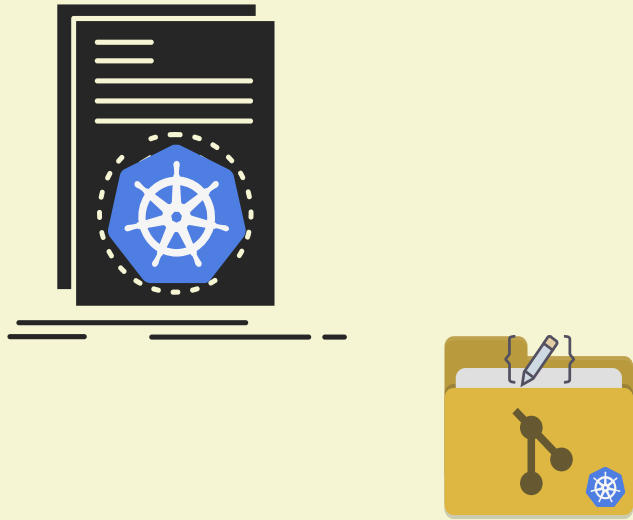
Kubernetes: E2E Test



Kubernetes: E2E Test



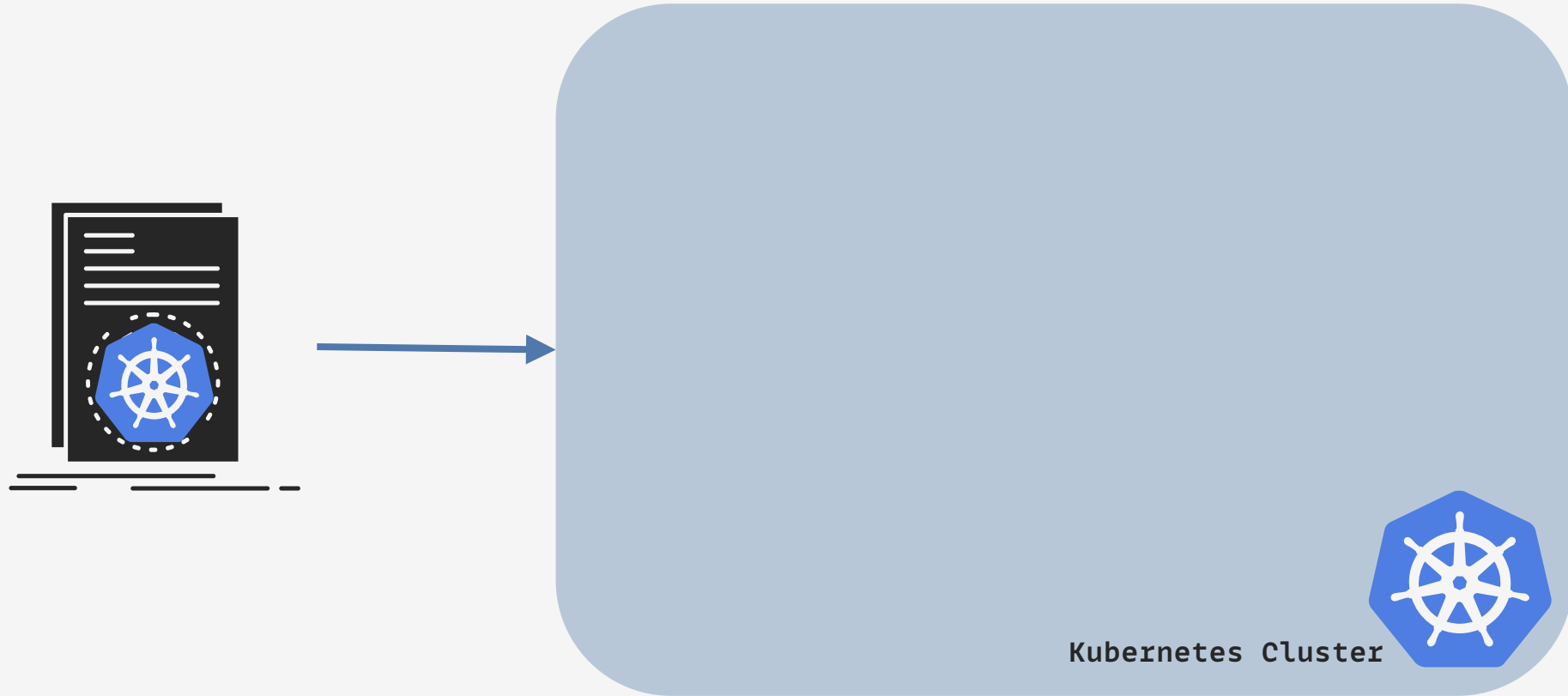
Kubernetes: E2E Test



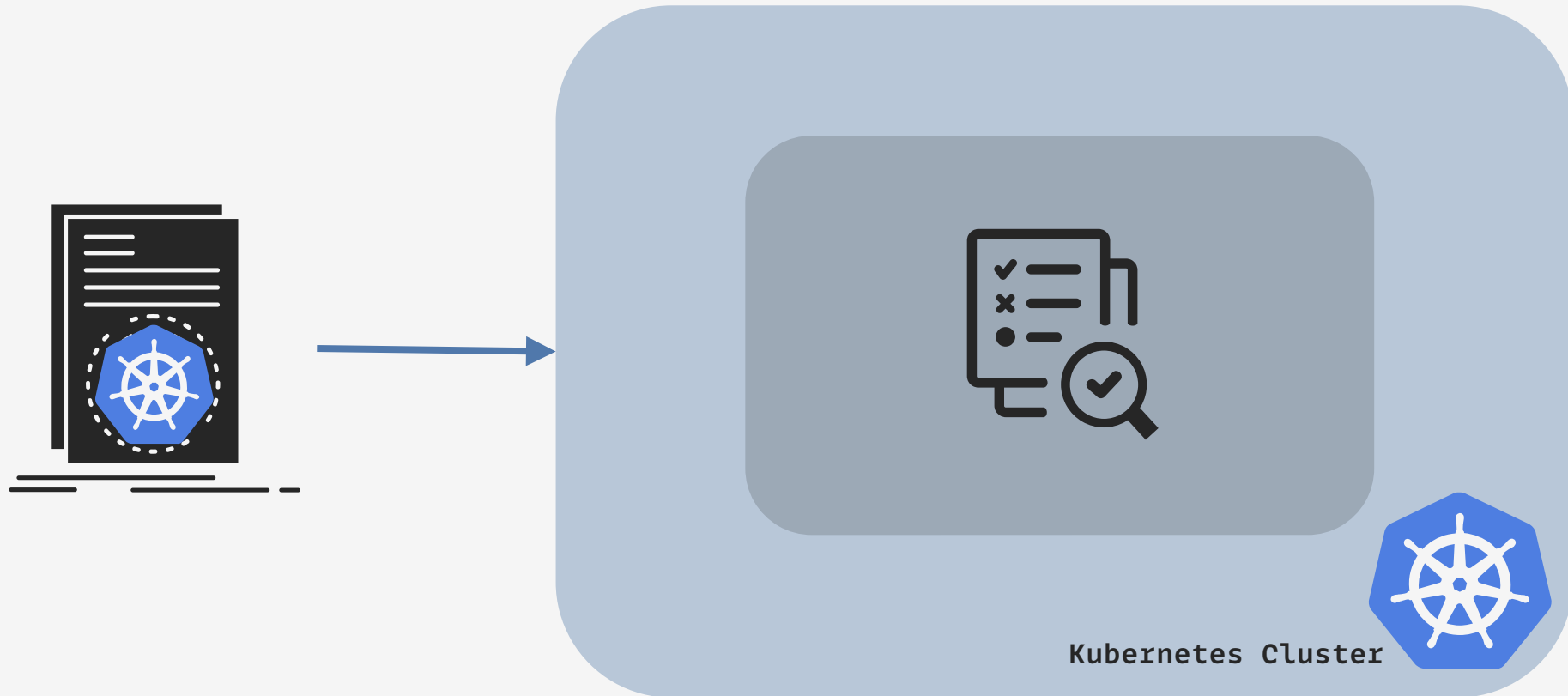
Kubernetes: E2E Test



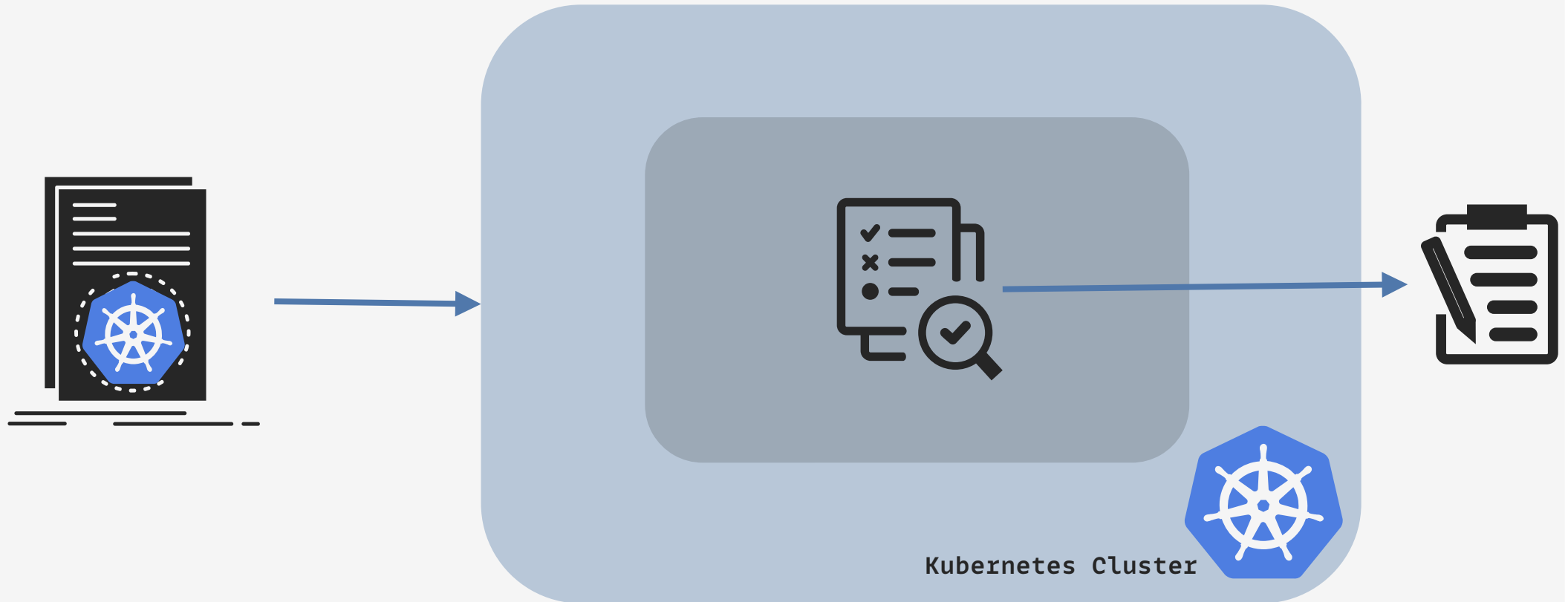
Kubernetes: E2E Test



Kubernetes: E2E Test



Kubernetes: E2E Test



Kubernetes: E2E Test



Kubernetes: E2E Test



Kubetest2!

What is Kubectl?

What is Kubetest2?

**A framework for deploying Kubernetes clusters,
and executing E2E (end-to-end) tests on them**

What is Kubectl?

Cluster Configuration

What is Kubetest2?

Cluster Configuration

E2E Testing, & Log Collection

What is Kubetest2?

Cluster Configuration

E2E Testing, & Log Collection

Test Environment Disposal

Kubetest2 (workflow)

Kubetest2 (workflow)

Cluster
Configuration

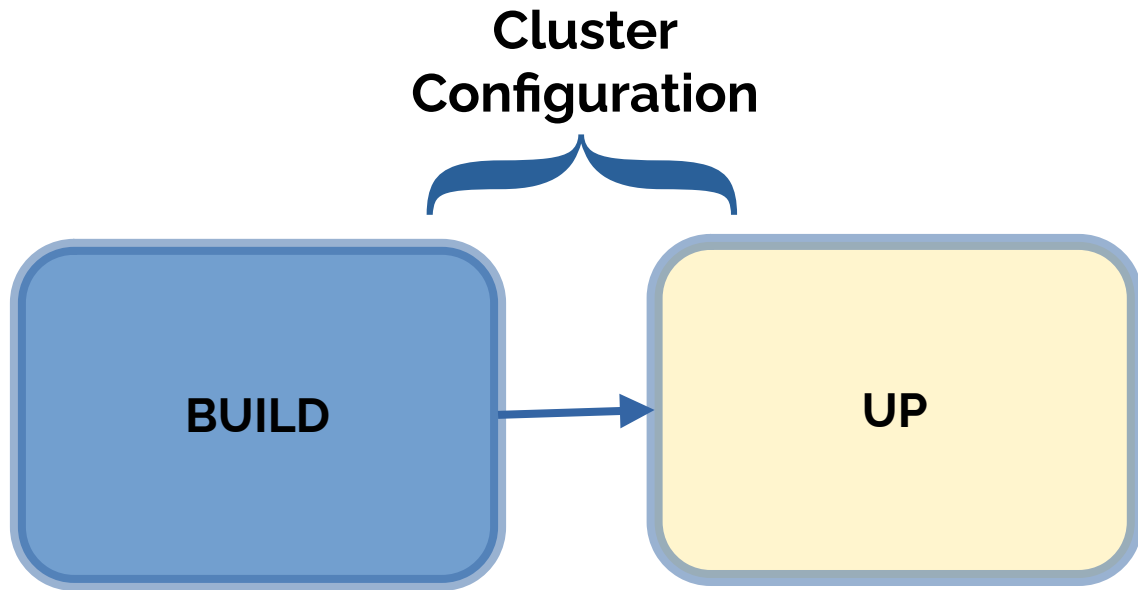
Kubetest2 (workflow)

Cluster
Configuration

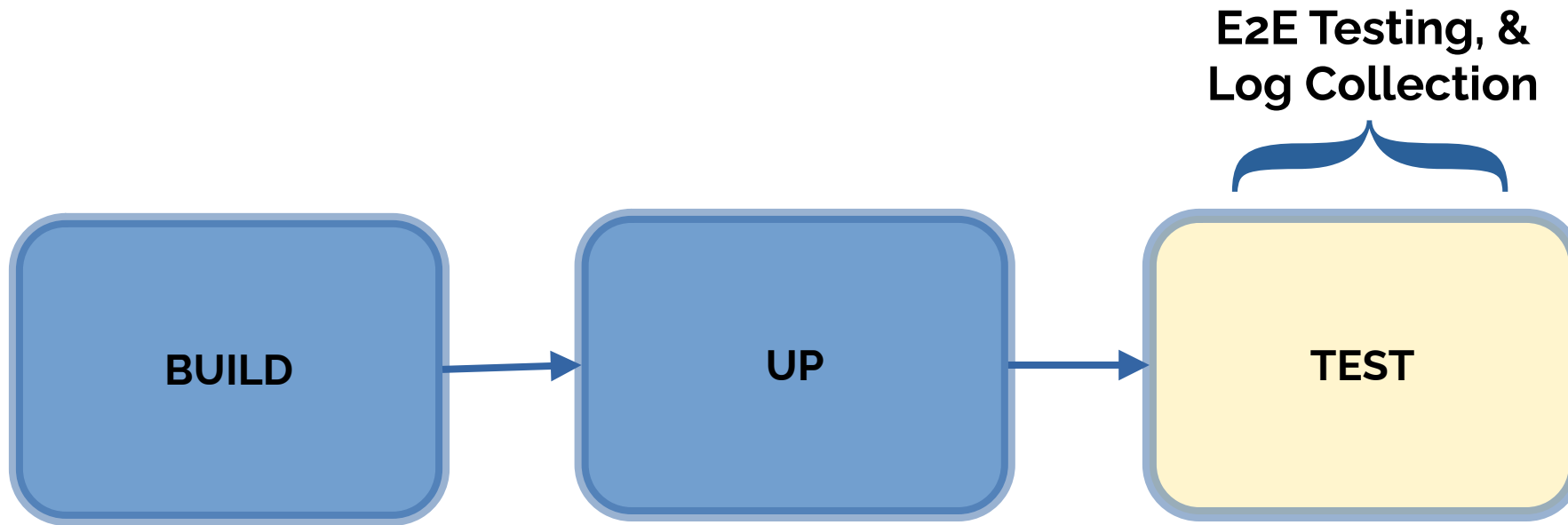


BUILD

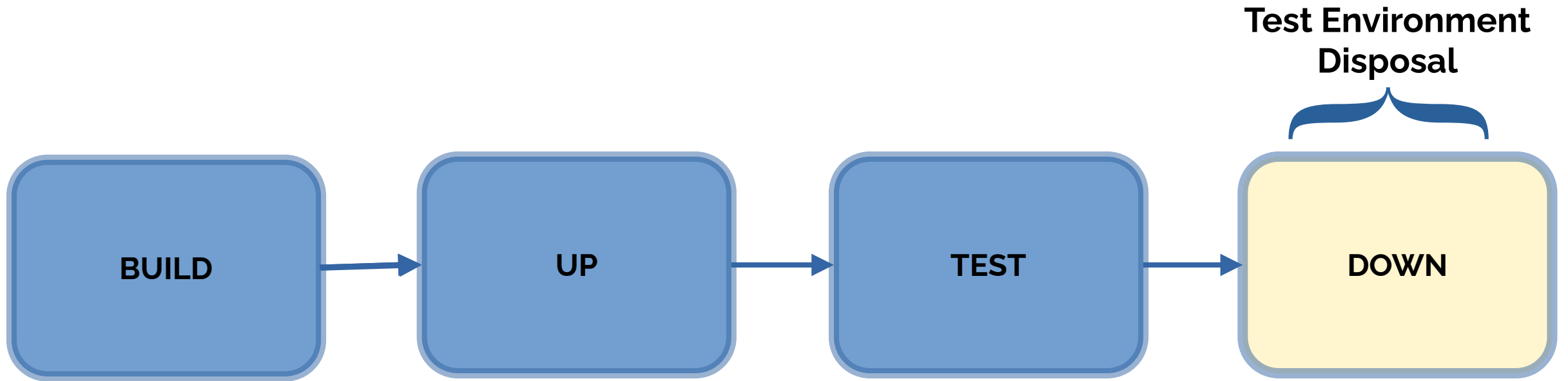
Kubetest2 (workflow)



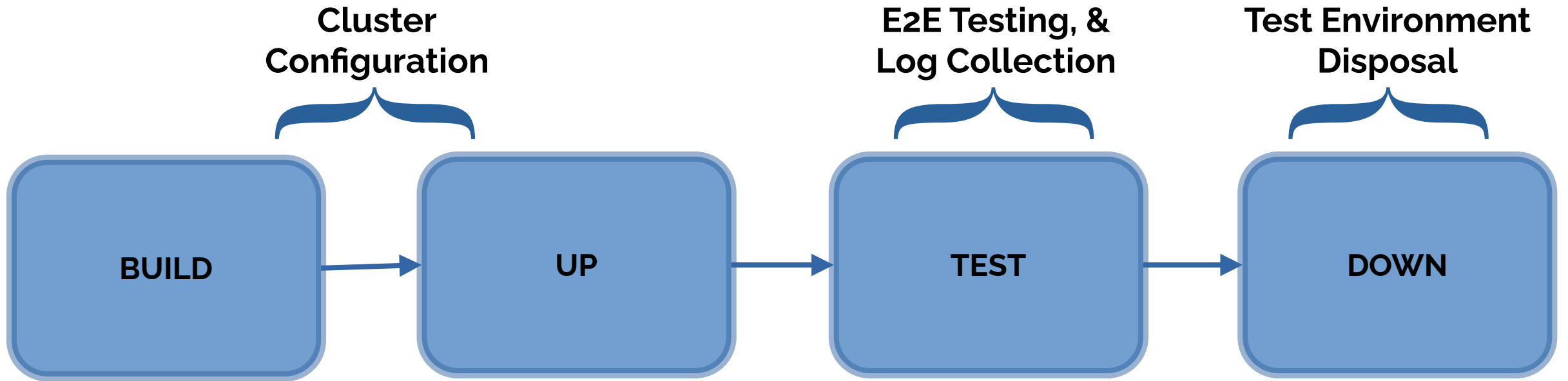
Kubetest2 (workflow)



Kubetest2 (workflow)



Kubetest2 (workflow)



Architecture of Kubetest2

Architecture of Kubetest2

**Kubetest2 is split into
*three independent executables***

Architecture of Kubectl2

Main Binary



Kubetest2

Architecture of Kubetest2

```
$ kubetest2
```

kubetest2 is a tool **for** kubernetes end to end testing.

It orchestrates creating clusters, building kubernetes, deleting clusters, running tests, etc.

kubetest2 should be called with a deployer like: 'kubetest2 kind --help'

For more information see: <https://github.com/kubernetes-sigs/kubetest2>

Usage:

kubetest2 [deployer] [flags]

Detected Deployers:

Detected Testers:

Architecture of Kubectl2

Main Binary

Deployer(s)

Kubetest2

Kubetest2
-DEPLOYER-1



Architecture of Kubetest2

```
$ kubetest2
```

kubetest2 is a tool **for** kubernetes end to end testing.

It orchestrates creating clusters, building kubernetes, deleting clusters, running tests, etc.

kubetest2 should be called with a deployer like: 'kubetest2 kind **--help**'

For more information see: <https://github.com/kubernetes-sigs/kubetest2>

Usage:

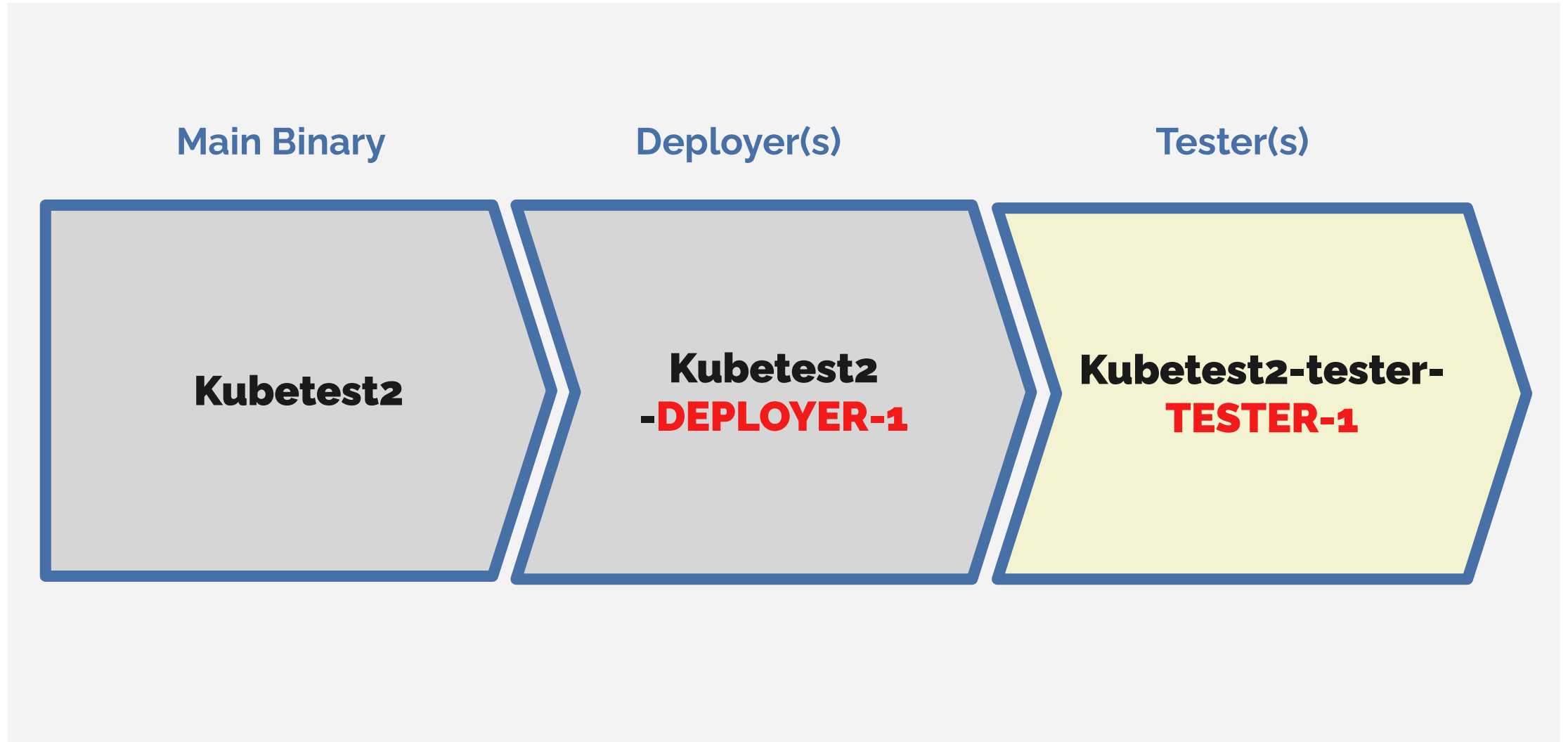
```
kubetest2 [deployer] [flags]
```

Detected Deployers:

DEPLOYER-1

Detected Testers:

Architecture of Kubectl2



Architecture of Kubetest2

```
$ kubetest2
```

kubetest2 is a tool **for** kubernetes end to end testing.

It orchestrates creating clusters, building kubernetes, deleting clusters, running tests, etc.

kubetest2 should be called with a deployer like: 'kubetest2 kind --help'

For more information see: <https://github.com/kubernetes-sigs/kubetest2>

Usage:

```
kubetest2 [deployer] [flags]
```

Detected Deployers:

```
DEPLOYER-1
```

Detected Testers:

```
TESTER-1
```

Architecture of Kubectl2



Architecture of Kubetest2

```
$ kubetest2
```

```
...
```

Usage:

```
kubetest2 [deployer] [flags]
```

Detected Deployers:

```
DEPLOYER-1
```

```
DEPLOYER-2
```

```
DEPLOYER-3
```

Detected Testers:

```
TESTER-1
```

```
TESTER-2
```

```
TESTER-3
```

Kubetest2 (Example)

Syntax

```
$ kubetest2 <deployer-name> \  
--up \  
--down \  
--test <tester> <test- arguments>
```

Kubetest2 (Example)

Syntax

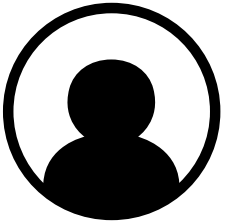
```
$ kubetest2 <deployer-name> \  
--up \  
--down \  
--test <tester> <test-arguments>
```

Example: upstream CNCF kubernetes test against a GKE cluster

```
$ kubetest2 gke\  
--up \  
--down \  
--test ginkgo -- --focus-regex "[Conformance]"
```

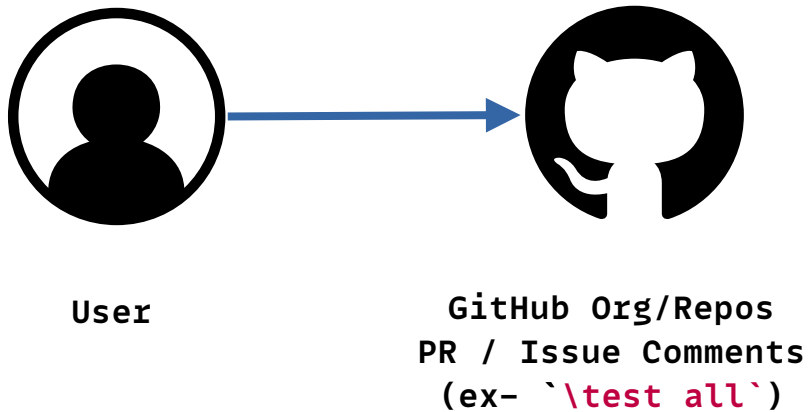
How the upstream Kubernetes Project integrates kubetest2 with Prow

How the upstream Kubernetes Project integrates kubetest2 with Prow

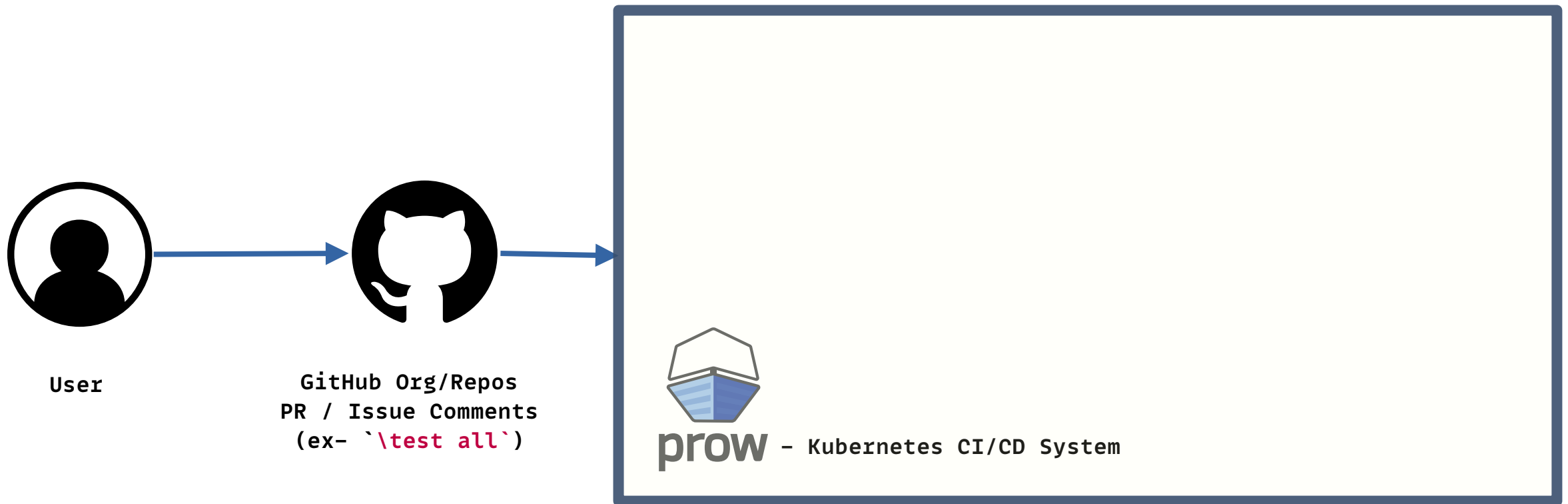


User

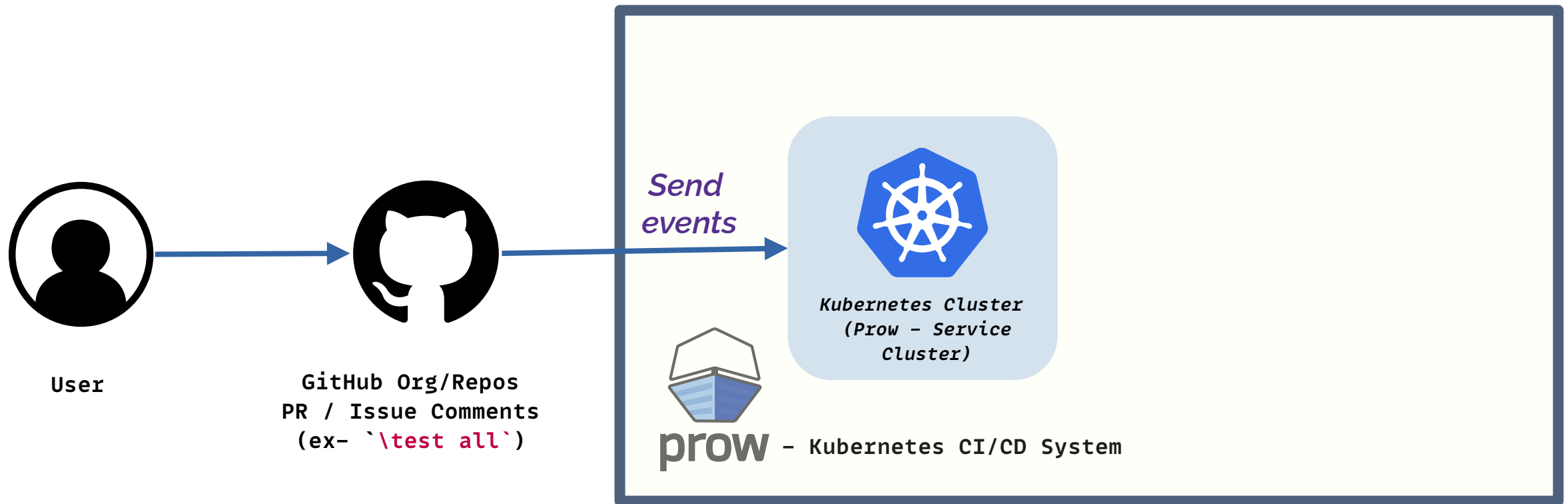
How the upstream Kubernetes Project integrates kubetest2 with Prow



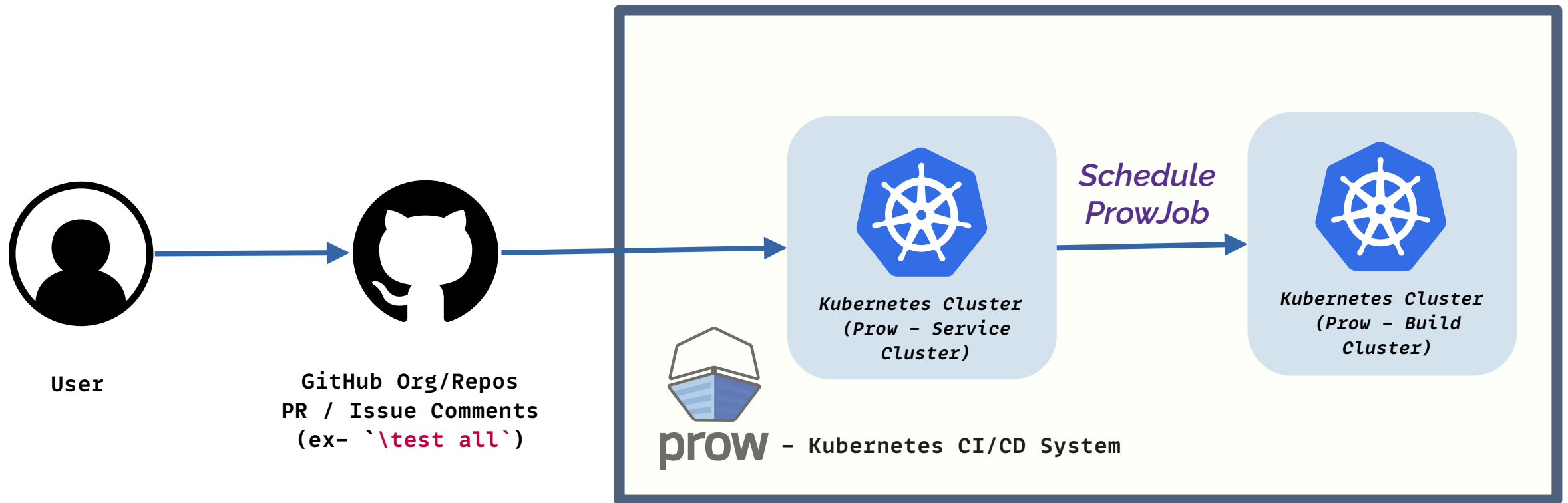
How the upstream Kubernetes Project integrates kubetest2 with Prow



How the upstream Kubernetes Project integrates kubetest2 with Prow



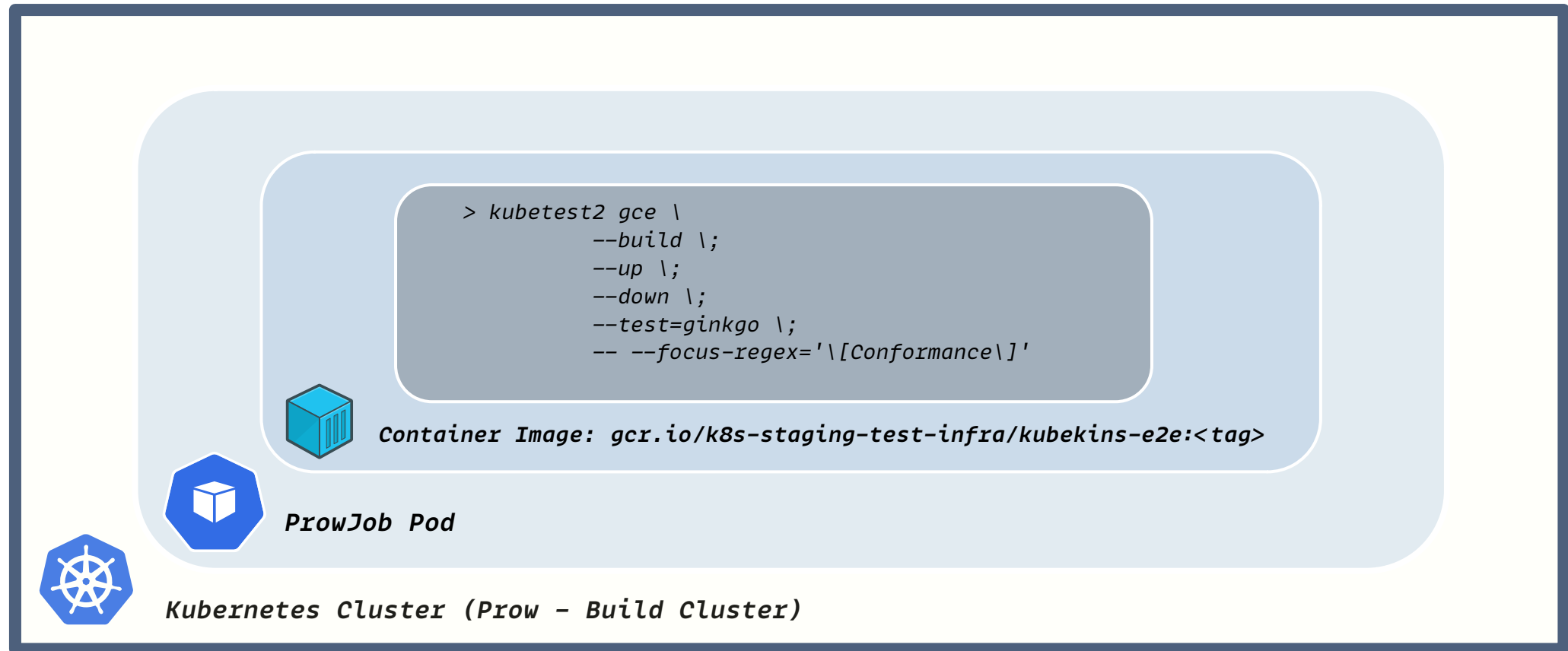
How the upstream Kubernetes Project integrates kubetest2 with Prow



How the upstream Kubernetes Project integrates kubetest2 with Prow



How the upstream Kubernetes Project integrates kubetest2 with Prow



Kubetest2: Primary Features

Consistent cluster life-cycle

Kubetest2: Primary Features

Consistent cluster life-cycle

Decoupled implementation of deployers, and
plug-&-play testers

Kubetest2: Primary Features

Consistent cluster life-cycle

Decoupled implementation of deployers, and
plug-&-play testers

Reproducible CI & local testing experience

Kubetest2: Primary Features

Consistent cluster life-cycle

Decoupled implementation of deployers, and
plug-&-play testers

Reproducible CI & local testing experience

Support for Boskos

Bespoke Deployer: Why?

At present, Kubetest2 *only* supports GCP, GKE, and KinD Deployers (in-tree)

Bespoke Deployer: Why?

At present, Kubetest2 *only* supports GCP, GKE, and KinD Deployers (in-tree)

but *enables* writing *Custom Deployers* for different cloud platforms *out-of-tree*

Demo!

Demo!

**Writing *Custom Deployer* for Kubectl2:
*AKS (Azure Kubernetes Services)***

Demo: Custom AKS Deployer

```
type Deployer interface {  
    Up() error  
    Down() error  
    IsUp() (up bool, err error)  
    DumpClusterLogs() error  
    Build() error //optional!  
}
```


Demo: Custom AKS Deployer

```
type Deployer interface {  
    Up() error  
    Down() error  
    IsUp() (up bool, err error)  
    DumpClusterLogs() error  
    Build() error //optional!  
}
```

Deployer defines the interface between Kubectl & a deployer

Demo: Custom AKS Deployer

```
type Deployer interface {  
    Up() error  
    Down() error  
    IsUp() (up bool, err error)  
    DumpClusterLogs() error  
    Build() error //optional!  
}
```

*Up should provision a
new cluster for testing*

Demo: Custom AKS Deployer

```
type Deployer interface {  
    Up() error  
    Down() error  
    IsUp() (up bool, err error)  
    DumpClusterLogs() error  
    Build() error //optional!  
}
```

*Down should tear
down the test cluster
if any*

Demo: Custom AKS Deployer

```
type Deployer interface {  
    Up() error  
    Down() error  
    IsUp() (up bool, err error)  
    DumpClusterLogs() error  
    Build() error //optional!  
}
```

*IsUp should return
true if a test cluster is
successfully provisioned*

Demo: Custom AKS Deployer

```
type Deployer interface {  
    Up() error  
    Down() error  
    IsUp() (up bool, err error)  
    DumpClusterLogs() error  
    Build() error //optional!  
}
```

*DumpClusterLogs
exports logs from the
cluster during Test Run*

Demo: Custom AKS Deployer

```
type Deployer interface {  
    Up() error  
    Down() error  
    IsUp() (up bool, err error)  
    DumpClusterLogs() error  
    Build() error //optional!  
}
```

*Build should build
Kubernetes & package
it in whatever format
the deployer consumes*

Demo: Custom AKS Deployer

Demo: Custom AKS Deployer

```
$ git clone https://github.com/kubernetes-sigs/kubetest2.git
```

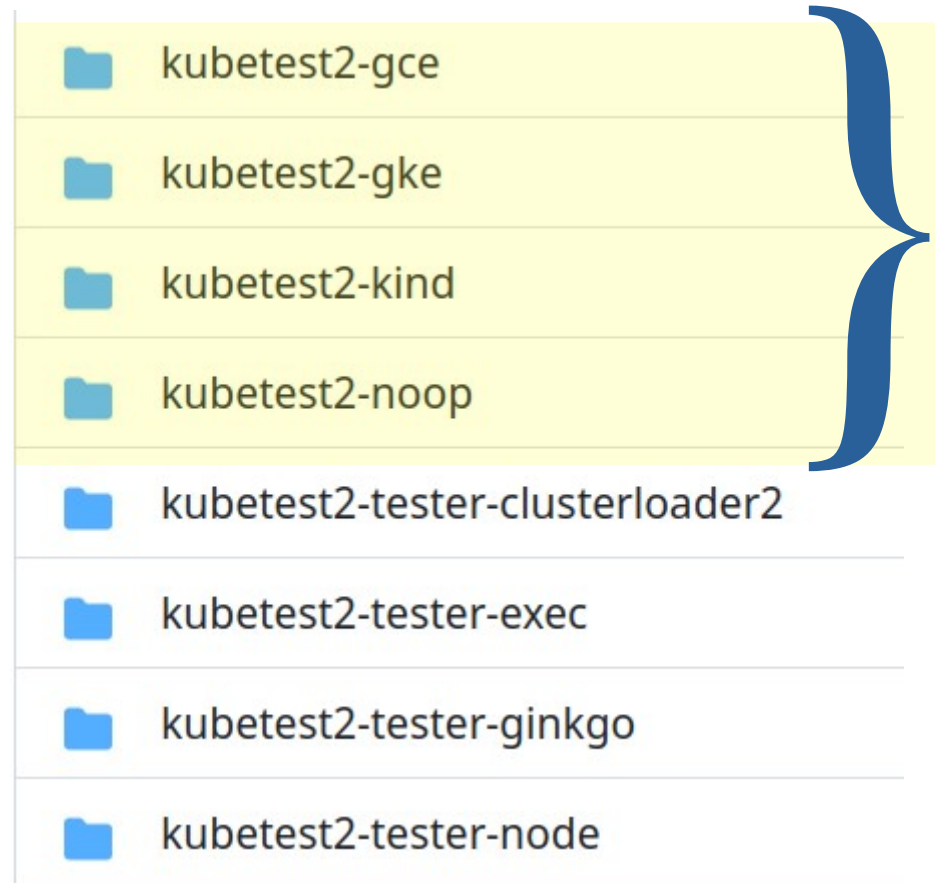
```
$ cd kubetest2
```

```
$ make install-all
```

*Install Kubetest2, and
all deployers & testers
in the PATH*

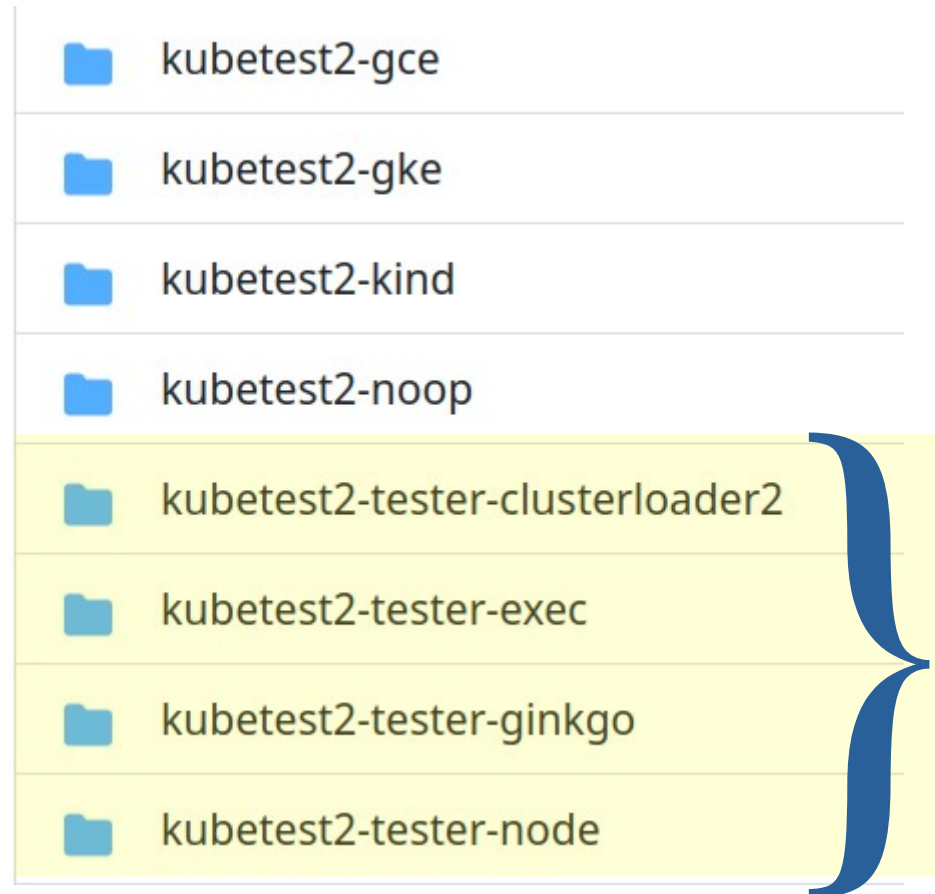


Demo: Custom AKS Deployer



Kubetest2-
DEPLOYER

Demo: Custom AKS Deployer



Kubetest2-tester-
DEPLOYER

Demo: Custom AKS Deployer

```
$ kubetest2
```

```
...
```

Usage:

```
kubetest2 [deployer] [flags]
```

Detected Deployers:

gce

gke

kind

noop

Detected Testers:

clusterloader-2

exec

ginkgo

node

Demo: Custom AKS Deployer

```
$ mkdir kubetest2-aks
```



*For new AKS Deployer,
in the format of
Kubetest2-DEPLOYER*

```
$ cd kubetest2-aks
```

```
// tree of kubetest2-aks
```

```
.
```

```
|-- deployer
```

```
|   |-- deployer.go
```

```
|-- main.go
```

```
`-- script
```

```
    |-- kube-down.sh
```

```
    |-- kube-up.sh
```

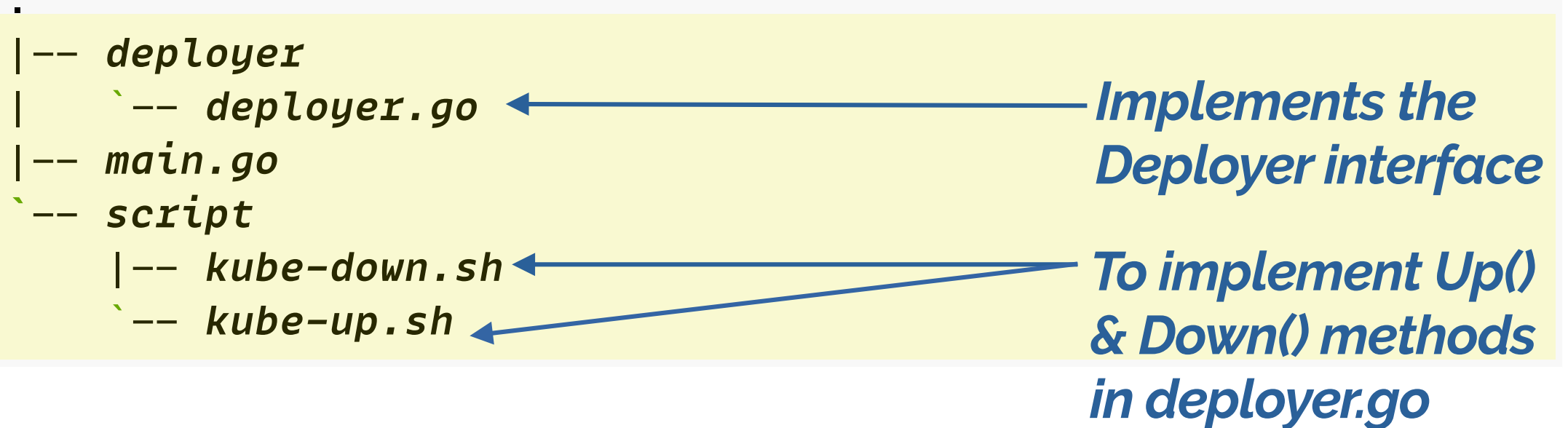
Demo: Custom AKS Deployer

```
$ mkdir kubetest2-aks
```

```
$ cd kubetest2-aks
```

```
// tree of kubetest2-aks
```

```
.  
|-- deployer  
|   |-- deployer.go ← Implements the  
|-- main.go Deployer interface  
|-- script  
    |-- kube-down.sh ← To implement Up()  
    |-- kube-up.sh ← & Down() methods  
                        in deployer.go
```



Demo: Custom AKS Deployer

```
$ mkdir kubetest2-aks
```

```
$ cd kubetest2-aks
```

```
// tree of kubetest2-aks
```

```
.  
|-- deployer  
|   |-- deployer.go  
|-- main.go ← Entrypoint!  
|-- script  
    |-- kube-down.sh  
    |-- kube-up.sh
```

Demo: `scripts/kube-up.sh`

```

1  #!/usr/bin/env bash
2  set -x
3  ## Check for required commands
4  command -v az > /dev/null || { echo "'az' command not not found" 1>&2; exit 1; }
5  command -v jq > /dev/null || { echo "'jq' command not not found" 1>&2; exit 1; }
6  ## Default variables
7  AZ_VM_SIZE=${AZ_VM_SIZE:-Standard_DS2_v2}
8  KUBECONFIG=${KUBECONFIG:-$HOME/.kube/${AZ_CLUSTER_NAME}.yaml}
9  ## Check for required variables
10 [[ -z "${AZ_RESOURCE_GROUP}" ]] && { echo 'AZ_RESOURCE_GROUP not specified. Aborting' 1>&2 ; exit 1; }
11 [[ -z "${AZ_CLUSTER_NAME}" ]] && { echo 'AZ_CLUSTER_NAME not specified. Aborting' 1>&2 ; exit 1; }
12 ## Create the resource group (idempotently)
13 if ! az group list | jq '.[].name' -r | grep -q ${AZ_RESOURCE_GROUP}; then
14     [[ -z "${AZ_LOCATION}" ]] && { echo 'AZ_LOCATION not specified. Aborting' 1>&2 ; exit 1; }
15     az group create --name=${AZ_RESOURCE_GROUP} --location=${AZ_LOCATION}
16 else
17     echo "'${AZ_RESOURCE_GROUP}' resource group is already created, skipping."
18 fi

```



```

1  #!/usr/bin/env bash
2  set -x
3  ## Check for required commands
4  command -v az > /dev/null || { echo "'az' command not not found" 1>&2; exit 1; }
5  command -v jq > /dev/null || { echo "'jq' command not not found" 1>&2; exit 1; }
6  ## Default variables
7  AZ_VM_SIZE=${AZ_VM_SIZE:-Standard_DS2_v2}
8  KUBECONFIG=${KUBECONFIG:-$HOME/.kube/${AZ_CLUSTER_NAME}.yaml}
9  ## Check for required variables
10 [[ -z "${AZ_RESOURCE_GROUP}" ]] && { echo 'AZ_RESOURCE_GROUP not specified. Aborting' 1>&2 ; exit 1; }
11 [[ -z "${AZ_CLUSTER_NAME}" ]] && { echo 'AZ_CLUSTER_NAME not specified. Aborting' 1>&2 ; exit 1; }
12 ## Create the resource group (idempotently)
13 if ! az group list | jq '.[].name' -r | grep -q ${AZ_RESOURCE_GROUP}; then
14     [[ -z "${AZ_LOCATION}" ]] && { echo 'AZ_LOCATION not specified. Aborting' 1>&2 ; exit 1; }
15     az group create --name=${AZ_RESOURCE_GROUP} --location=${AZ_LOCATION}
16 else
17     echo "'${AZ_RESOURCE_GROUP}' resource group is already created, skipping."
18 fi

```

```
19 ## create aks cluster if resource group was created
20 if az group list | jq '.[].name' -r | grep -q ${AZ_RESOURCE_GROUP}; then
21     ## check if AKS cluster was already created
22     if ! az aks list | jq '.[].name' -r | grep -q ${AZ_CLUSTER_NAME}; then
23         echo "Creating '${AZ_CLUSTER_NAME}' Kubernetes cluster"
24         az aks create \
25             --resource-group ${AZ_RESOURCE_GROUP} \
26             --name ${AZ_CLUSTER_NAME} \
27             --generate-ssh-keys \
28             --vm-set-type VirtualMachineScaleSets \
29             --node-vm-size ${AZ_VM_SIZE} \
30             --load-balancer-sku standard \
31             --enable-managed-identity \
32             --node-count 3 \
33             --zones 1 2 3
34     else
35         echo "'${AZ_CLUSTER_NAME}' Kubernetes cluster is already created, skipping."
36     fi
```

```
37  ## create KUBECONFIG so that cluster can be accessed using existing login credentials
38  if az aks list | jq '.[].name' -r | grep -q ${AZ_CLUSTER_NAME}; then
39      ## Azure ignores KUBECONFIG, but we can specify with --file flag
40      az aks get-credentials \
41          --resource-group ${AZ_RESOURCE_GROUP} \
42          --name ${AZ_CLUSTER_NAME} \
43          --file ${KUBECONFIG}
44  fi
45  fi
```

Demo: `scripts/kube-down.sh`

```

1  #!/usr/bin/env bash
2
3  ## Check for required commands
4  command -v az > /dev/null || { echo "'az' command not not found" 1>&2; exit 1; }
5  command -v jq > /dev/null || { echo "'jq' command not not found" 1>&2; exit 1; }
6  ## Check for required variables
7  [[ -z "$AZ_RESOURCE_GROUP" ]] && { echo 'AZ_RESOURCE_GROUP not specified. Aborting' 1>&2 ; exit 1; }
8  [[ -z "$AZ_CLUSTER_NAME" ]] && { echo 'AZ_CLUSTER_NAME not specified. Aborting' 1>&2 ; exit 1; }
9  ## delete aks cluster if resource group was created
10 if az group list | jq '.[].name' -r | grep -q "^${AZ_RESOURCE_GROUP}$"; then
11     if az aks list | jq '.[].name' -r | grep -q "^${AZ_CLUSTER_NAME}$"; then
12         az aks delete \
13             --resource-group "${AZ_RESOURCE_GROUP}" \
14             --name "${AZ_CLUSTER_NAME}" --yes
15     else
16         echo "Cannot find '$AZ_CLUSTER_NAME' Kubernetes cluster, skipping."
17     fi
18 fi

```

```

1  #!/usr/bin/env bash
2
3  ## Check for required commands
4  command -v az > /dev/null || { echo "'az' command not not found" 1>&2; exit 1; }
5  command -v jq > /dev/null || { echo "'jq' command not not found" 1>&2; exit 1; }
6  ## Check for required variables
7  [[ -z "$AZ_RESOURCE_GROUP" ]] && { echo 'AZ_RESOURCE_GROUP not specified. Aborting' 1>&2 ; exit 1; }
8  [[ -z "$AZ_CLUSTER_NAME" ]] && { echo 'AZ_CLUSTER_NAME not specified. Aborting' 1>&2 ; exit 1; }
9  ## delete aks cluster if resource group was created
10 if az group list | jq '.[].name' -r | grep -q "^${AZ_RESOURCE_GROUP}$"; then
11     if az aks list | jq '.[].name' -r | grep -q "^${AZ_CLUSTER_NAME}$"; then
12         az aks delete \
13             --resource-group "${AZ_RESOURCE_GROUP}" \
14             --name "${AZ_CLUSTER_NAME}" --yes
15     else
16         echo "Cannot find '$AZ_CLUSTER_NAME' Kubernetes cluster, skipping."
17     fi
18 fi

```

Demo: `deployer/deployer.go`

```
package deployer
import (
    ...
    "sigs.k8s.io/kubetest2/pkg/types"
    ...
)
```

```
var (
    GitTag string
    randomPostFix, _ = RandString(6)
    aksResourceGroup = "aks-rg-" + randomPostFix
    aksClusterName = "aks-cluster-" + randomPostFix
)
```

```
// Name is the name of the deployer
const Name = "aks"
```

```
func (d *deployer) buildEnv() []string {
    env := os.Environ()
    env = append(env, fmt.Sprintf("AZ_LOCATION=%s", "westus2"))
    env = append(env, fmt.Sprintf("AZ_RESOURCE_GROUP=%s", aksResourceGroup))
    env = append(env, fmt.Sprintf("AZ_CLUSTER_NAME=%s", aksClusterName))
    env = append(env, fmt.Sprintf("HOME=%s", os.UserHomeDir()))
    env = append(env, fmt.Sprintf("KUBECONFIG=%s", filepath.Join(home, ".kube", aksClusterName+".yaml")))
    return env
}
...
```



```
package deployer
import (
    ...
    "sigs.k8s.io/kubetest2/pkg/types"
    ...
)
```

```
var (
    GitTag string
    randomPostFix, _ = RandString(6)
    aksResourceGroup = "aks-rg-" + randomPostFix
    aksClusterName = "aks-cluster-" + randomPostFix
)
```

```
// Name is the name of the deployer
```

```
const Name = "aks"
```

```
func (d *deployer) buildEnv() []string {
    env := os.Environ()
    env = append(env, fmt.Sprintf("AZ_LOCATION=%s", "westus2"))
    env = append(env, fmt.Sprintf("AZ_RESOURCE_GROUP=%s", aksResourceGroup))
    env = append(env, fmt.Sprintf("AZ_CLUSTER_NAME=%s", aksClusterName))
    env = append(env, fmt.Sprintf("HOME=%s", os.UserHomeDir()))
    env = append(env, fmt.Sprintf("KUBECONFIG=%s", filepath.Join(home, ".kube", aksClusterName+".yaml")))
    return env
}
```

```
...
```

```

package deployer
import (
    ...
    "sigs.k8s.io/kubetest2/pkg/types"
    ...
)

var (
    GitTag string
    randomPostFix, _ = RandString(6)
    aksResourceGroup = "aks-rg-" + randomPostFix
    aksClusterName = "aks-cluster-" + randomPostFix
)

```

```

// Name is the name of the deployer
const Name = "aks"

```

```

func (d *deployer) buildEnv() []string {
    env := os.Environ()
    env = append(env, fmt.Sprintf("AZ_LOCATION=%s", "westus2"))
    env = append(env, fmt.Sprintf("AZ_RESOURCE_GROUP=%s", aksResourceGroup))
    env = append(env, fmt.Sprintf("AZ_CLUSTER_NAME=%s", aksClusterName))
    env = append(env, fmt.Sprintf("HOME=%s", os.UserHomeDir()))
    env = append(env, fmt.Sprintf("KUBECONFIG=%s", filepath.Join(home, ".kube", aksClusterName+".yaml")))
    return env
}

...

```

```

package deployer
import (
    ...
    "sigs.k8s.io/kubetest2/pkg/types"
    ...
)

var (
    GitTag string
    randomPostFix, _ = RandString(6)
    aksResourceGroup = "aks-rg-" + randomPostFix
    aksClusterName = "aks-cluster-" + randomPostFix
)

// Name is the name of the deployer
const Name = "aks"

```

```

func (d *deployer) buildEnv() []string {
    env := os.Environ()
    env = append(env, fmt.Sprintf("AZ_LOCATION=%s", "westus2"))
    env = append(env, fmt.Sprintf("AZ_RESOURCE_GROUP=%s", aksResourceGroup))
    env = append(env, fmt.Sprintf("AZ_CLUSTER_NAME=%s", aksClusterName))
    env = append(env, fmt.Sprintf("HOME=%s", os.UserHomeDir()))
    env = append(env, fmt.Sprintf("KUBECONFIG=%s", filepath.Join(home, ".kube", aksClusterName+".yaml")))
    return env
}

```

...

...

```
type deployer struct {  
    commonOptions    types.Options  
    logsDir           string  
    overwriteLogsDir bool  
    repoRoot          string  
    kubeconfigPath   string  
}
```

// New implements deployer.New

```
func New(opts types.Options) (types.Deployer, *pflag.FlagSet) {  
    d := &deployer{  
        commonOptions:    opts,  
        logsDir:           filepath.Join(opts.RunDir(), "cluster-logs"),  
        overwriteLogsDir:   false,  
        repoRoot:           "",  
        kubeconfigPath:    "",  
    }  
  
    // bindFlags() - helper to create & bind a flagset to the deployer  
    return d, bindFlags(d)  
}
```

// assert that New implements types.NewDeployer

```
var _ types.NewDeployer = New
```

...

...

```
type deployer struct {  
    commonOptions types.Options  
    logsDir        string  
    overwriteLogsDir bool  
    repoRoot        string  
    kubeconfigPath string  
}
```

// New implements deployer.New

```
func New(opts types.Options) (types.Deployer, *pflag.FlagSet) {  
    d := &deployer{  
        commonOptions: opts,  
        logsDir:        filepath.Join(opts.RunDir(), "cluster-logs"),  
        overwriteLogsDir: false,  
        repoRoot:        "",  
        kubeconfigPath: "",  
    }  
  
    // bindFlags() - helper to create & bind a flagset to the deployer  
    return d, bindFlags(d)  
}
```

// assert that New implements types.NewDeployer

```
var _ types.NewDeployer = New
```

...

```

...

type deployer struct {
    commonOptions    types.Options
    logsDir           string
    overwriteLogsDir bool
    repoRoot          string
    kubeconfigPath   string
}

// New implements deployer.New
func New(opts types.Options) (types.Deployer, *pflag.FlagSet) {
    d := &deployer{
        commonOptions:    opts,
        logsDir:           filepath.Join(opts.RunDir(), "cluster-logs"),
        overwriteLogsDir:  false,
        repoRoot:          "",
        kubeconfigPath:   "",
    }

    // bindFlags() - helper to create & bind a flagset to the deployer
    return d, bindFlags(d)
}

// assert that New implements types.NewDeployer
var _ types.NewDeployer = New

```

...

...

```
func (d *deployer) IsUp() (bool, error) {
    klog.V(1).Info("AKS deployer starting IsUp()")

    env := d.buildEnv()

    cmd := exec.Command("/usr/local/bin/kubectl", "get", "nodes",
                        "-o=name", "--kubeconfig="+filepath.Join(homeDir(), ".kube", aksClusterName+".yaml"))

    cmd.SetEnv(env...)
    cmd.SetStderr(os.Stderr)

    output, err := exec.Output(cmd)
    if err != nil {
        return false, fmt.Errorf("is up failed to get nodes: %w", err)
    }
    return len(output) > 0, nil
}
```

...

...

```
func (d *deployer) IsUp() (bool, error) {
    klog.V(1).Info("AKS deployer starting IsUp()")

    env := d.buildEnv()

    cmd := exec.Command("/usr/local/bin/kubectl", "get", "nodes",
                        "-o=name", "--kubeconfig="+filepath.Join(homeDir(), ".kube", aksClusterName+".yaml"))

    cmd.SetEnv(env...)
    cmd.SetStderr(os.Stderr)

    output, err := exec.Output(cmd)
    if err != nil {
        return false, fmt.Errorf("is up failed to get nodes: %w", err)
    }
    return len(output) > 0, nil
}
```

...

...

```
func (d *deployer) DumpClusterLogs() error {
    klog.V(1).Info("AKS deployer starting DumpClusterLogs()")

    if err := d.makeLogsDir(); err != nil {
        return fmt.Errorf("failed to make logs directory: %w", err)
    }

    // `kubectl cluster-info dump --kubeconfig=<path-to-kubeconfig>`
    if err := d.kubectlDump(); err != nil {
        return fmt.Errorf("failed to dump cluster info with kubectl: %w", err)
    }
    return nil
}
```

...

...

```
func (d *deployer) DumpClusterLogs() error {
    klog.V(1).Info("AKS deployer starting DumpClusterLogs()")

    if err := d.makeLogsDir(); err != nil {
        return fmt.Errorf("failed to make logs directory: %w", err)
    }

    // `kubectl cluster-info dump --kubeconfig=<path-to-kubeconfig>`
    if err := d.kubectlDump(); err != nil {
        return fmt.Errorf("failed to dump cluster info with kubectl: %w", err)
    }
    return nil
}
```

...

...

```
func (d *deployer) Up() error {
    klog.V(1).Info("AKS deployer starting Up()")

    env := d.buildEnv()

    defer func() {
        if err := d.DumpClusterLogs(); err != nil {
            klog.Warningf("Dumping cluster logs at the end of Up() failed: %s", err)
        }
    }()

    if err := d.runScript("kube-up.sh"); err != nil {
        return fmt.Errorf("error encountered during kube-up.sh: %w", err)
    }

    // `kubectl get nodes -o name --kubecfg=<path-to-kubecfg>`
    if isUp, err := d.IsUp(); err != nil {
        klog.Warningf("failed to check if cluster is up: %s", err)
    } else if isUp {
        klog.V(1).Infof("cluster reported as up")
    } else {
        klog.Errorf("cluster reported as down")
    }
    return nil
}
```

...

...

```
func (d *deployer) Up() error {
    klog.V(1).Info("AKS deployer starting Up()")

    env := d.buildEnv()

    defer func() {
        if err := d.DumpClusterLogs(); err != nil {
            klog.Warningf("Dumping cluster logs at the end of Up() failed: %s", err)
        }
    }()

    if err := d.runScript("kube-up.sh"); err != nil {
        return fmt.Errorf("error encountered during kube-up.sh: %w", err)
    }

    // `kubectl get nodes -o name --kubeconfig=<path-to-kubeconfig>`
    if isUp, err := d.IsUp(); err != nil {
        klog.Warningf("failed to check if cluster is up: %s", err)
    } else if isUp {
        klog.V(1).Infof("cluster reported as up")
    } else {
        klog.Errorf("cluster reported as down")
    }
    return nil
}
```

...

...

```
func (d *deployer) Down() error {
    klog.V(1).Info("AKS deployer starting Down()")

    if err := d.runScript("kube-down.sh"); err != nil {
        return fmt.Errorf("error encountered during kube-down.sh: %w", err)
    }
    return nil
}
```

```
func (d *deployer) Kubeconfig() (string, error) {

    if d.KubeconfigPath != "" {
        return d.KubeconfigPath, nil
    }
    if kconfig, ok := os.LookupEnv("KUBECONFIG"); ok {
        return kconfig, nil
    }
    home, err := os.UserHomeDir()
    if err != nil {
        return "", err
    }
    return filepath.Join(home, ".kube", "config"), nil
}
```

...

...

```
func (d *deployer) Down() error {
    klog.V(1).Info("AKS deployer starting Down()")

    if err := d.runScript("kube-down.sh"); err != nil {
        return fmt.Errorf("error encountered during kube-down.sh: %w", err)
    }
    return nil
}
```

```
func (d *deployer) Kubeconfig() (string, error) {

    if d.KubeconfigPath != "" {
        return d.KubeconfigPath, nil
    }
    if kconfig, ok := os.LookupEnv("KUBECONFIG"); ok {
        return kconfig, nil
    }
    home, err := os.UserHomeDir()
    if err != nil {
        return "", err
    }
    return filepath.Join(home, ".kube", "config"), nil
}
```

...

...

```
func (d *deployer) Down() error {
    klog.V(1).Info("AKS deployer starting Down()")

    if err := d.runScript("kube-down.sh"); err != nil {
        return fmt.Errorf("error encountered during kube-down.sh: %w", err)
    }
    return nil
}
```

```
func (d *deployer) Kubeconfig() (string, error) {
```

```
    if d.KubeconfigPath != "" {
        return d.KubeconfigPath, nil
    }
```

```
    if kconfig, ok := os.LookupEnv("KUBECONFIG"); ok {
        return kconfig, nil
    }
```

```
    home, err := os.UserHomeDir()
    if err != nil {
        return "", err
    }
```

```
    return filepath.Join(home, ".kube", "config"), nil
```

```
}
```

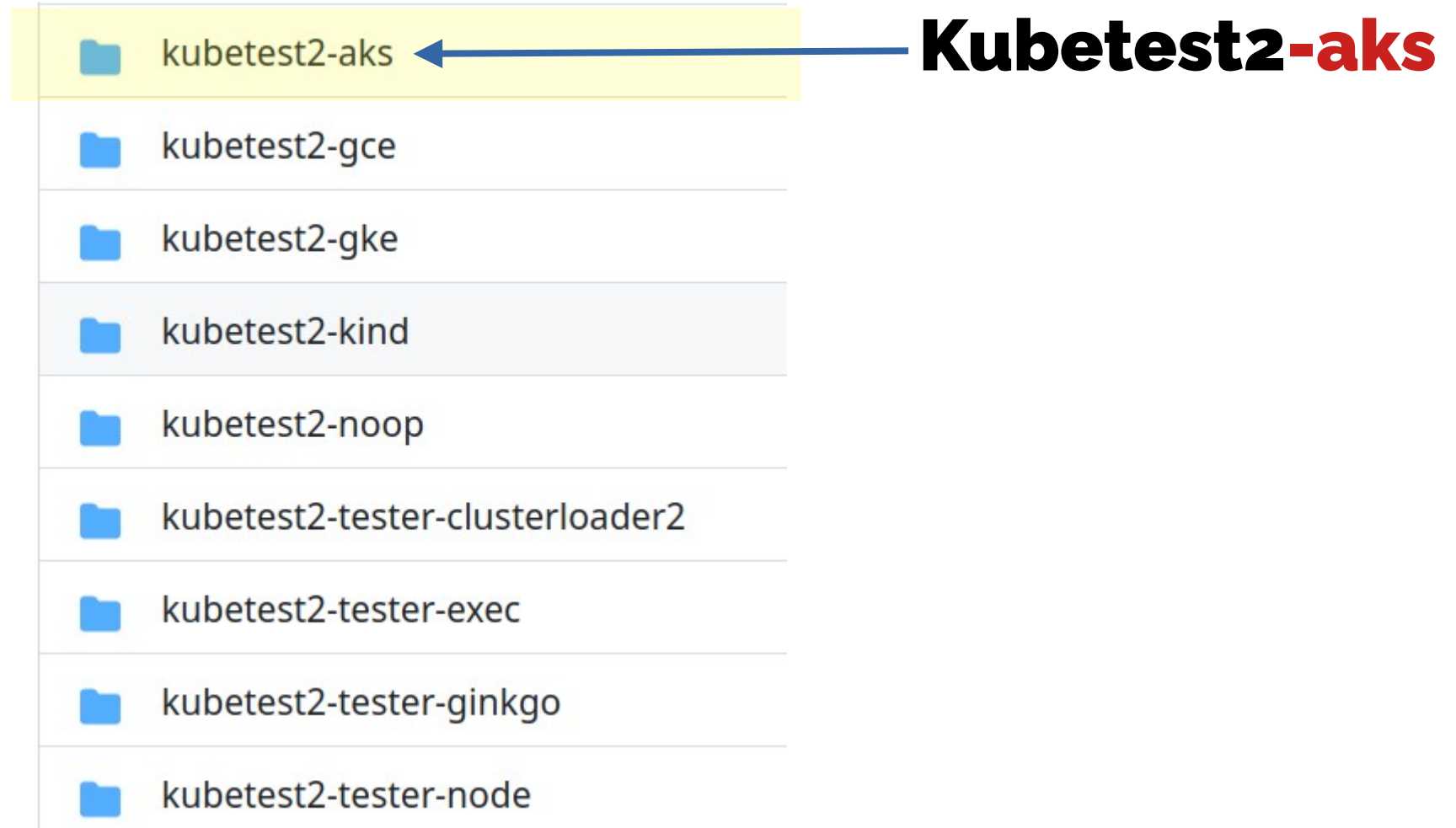
...

Demo: main.go


```
1 package main
2
3 import (
4     "sigs.k8s.io/kubetest2/pkg/app"
5     "sigs.k8s.io/kubetest2/kubetest2-aks/deployer"
6 )
7
8 func main() {
9     app.Main(deployer.Name, deployer.New)
10 }
```

```
1 package main
2
3 import (
4     "sigs.k8s.io/kubetest2/pkg/app"
5     "sigs.k8s.io/kubetest2/kubetest2-aks/deployer"
6 )
7
8 func main() {
9     app.Main(deployer.Name, deployer.New)
10 }
```

Demo: Custom AKS Deployer



Demo: Custom AKS Deployer



Demo: Custom AKS Deployer

`$ make install-all` ← *rerun to install new
Kubetest2-aks
deployer*

Demo: Custom AKS Deployer

```
$ kubetest2
```

```
...
```

Usage:

```
kubetest2 [deployer] [flags]
```

Detected Deployers:

aks

gce

gke

kind

noop

Detected Testers:

clusterloader-2

exec

ginkgo

node

Demo: Custom AKS Deployer

```
$ kubetest2 aks --up --down
```

```
psaggu@demo-vm:~/demo/kubetest2$ kubetest2 aks --up --down
```

```
I0910 18:09:52.198210 9651 app.go:61] The files in RunDir shall not be part of Artifacts
```

```
I0910 18:09:52.198341 9651 app.go:62] pass rundir-in-artifacts flag True for RunDir to be part of Artifacts
```

```
I0910 18:09:52.198357 9651 app.go:64] RunDir for this run: "/home/psaggu/demo/kubetest2/_rundir/bd9c5561-8323-4aad-bb07-4e4f8de562c7"
```

```
I0910 18:09:52.201880 9651 app.go:128] ID for this run: "bd9c5561-8323-4aad-bb07-4e4f8de562c7"
```

```
+ command -v az
```

```
+ command -v jq
```

```
+ AZ_VM_SIZE=Standard_DS2_v2
```

```
+ KUBECONFIG=/home/psaggu/.kube/aks-cluster-mfilrl.yaml
```

```
+ [[ -z aks-rg-mfilrl ]]
```

```
+ [[ -z aks-cluster-mfilrl ]]
```

```
+ az group list
```

```
+ grep -q aks-rg-mfilrl
```

```
+ jq '.[0].name' -r
```

```
+ [[ -z westus2 ]]
```

```
+ az group create --name=aks-rg-mfilrl --location=westus2
```

```
{
```

```
  "id": "/subscriptions/
```

```
  "location": "westus2",
```

```
  "managedBy": null,
```

```
  "name": "aks-rg-mfilrl",
```

```
  "properties": {
```

```
    "provisioningState": "Succeeded"
```

```
  },
```

```
  "tags": null,
```

```
  "type": "Microsoft.Resources/resourceGroups"
```

```
}
```

```
+ az group list
```

```
+ jq '.[0].name' -r
```

```
+ grep -q aks-rg-mfilrl
```

```
+ az aks list
```

```
+ jq '.[0].name' -r
```

```
+ grep -q aks-cluster-mfilrl
```

```
+ echo 'Creating \''aks-cluster-mfilrl\' Kubernetes cluster'
```

```
Creating 'aks-cluster-mfilrl' Kubernetes cluster
```

```
+ az aks create --resource-group aks-rg-mfilrl --name aks-cluster-mfilrl --generate-ssh-keys --vm-set-type VirtualMachineScaleSets --node-vm-size Standard_DS2_v2 --load-balancer-sku standard --enable-managed-identity --node-count 3 --zones 1 2 3
```

```
Running ..
```



```
]
},
"nodeResourceGroup": "MC_aks-rg-mfilrl_aks-cluster-mfilrl_westus2",
"podIdentityProfile": null,
"powerState": {
  "code": "Running"
},
"privateFqdn": null,
"privateLinkResources": null,
"provisioningState": "Succeeded",
"publicNetworkAccess": null,
"resourceGroup": "aks-rg-mfilrl",
"securityProfile": {
  "azureKeyVaultKms": null,
  "defender": null
},
"servicePrincipalProfile": {
  "clientId": "msi",
  "secret": null
},
"sku": {
  "name": "Basic",
  "tier": "Free"
},
"storageProfile": {
  "diskCsiDriver": {
    "enabled": true
  },
  "fileCsiDriver": {
    "enabled": true
  },
  "snapshotController": {
    "enabled": true
  }
},
"systemData": null,
"tags": null,
"type": "Microsoft.ContainerService/ManagedClusters",
"windowsProfile": null
}
```

```
+ az aks list
```

```
+ jq '.[].name' -r
```

```
+ grep -q aks-cluster-mfilrl
```

```
+ az aks get-credentials --resource-group aks-rg-mfilrl --name aks-cluster-mfilrl --file /home/psaggu/.kube/aks-cluster-mfilrl.yaml
```

```
Merged "aks-cluster-mfilrl" as current context in /home/psaggu/.kube/aks-cluster-mfilrl.yaml
```

```
psaggu@demo-vm:~/demo/kubetest2$ ls
```

File: _rundir/bd9c5561-8323-4aad-bb07-4e4f8de562c7/cluster-logs/cluster-info.log

```
1 {
2   "kind": "NodeList",
3   "apiVersion": "v1",
4   "metadata": {
5     "resourceVersion": "1456"
6   },
7   "items": [
8     {
9       "metadata": {
10        "name": "aks-nodepool1-1456",
11        "uid": "ff853e29-47d9-4bcf-9c3e-f710e0302307",
12        "resourceVersion": "1267",
13        "creationTimestamp": "2022-09-10T18:12:30Z",
14        "labels": {
15          "agentpool": "nodepool1",
16          "beta.kubernetes.io/arch": "amd64",
17          "beta.kubernetes.io/instance-type": "Standard_DS2_v2",
18          "beta.kubernetes.io/os": "linux",
19          "failure-domain.beta.kubernetes.io/region": "westus2",
20          "failure-domain.beta.kubernetes.io/zone": "westus2-1",
21          "kubernetes.azure.com/agentpool": "nodepool1",
22          "kubernetes.azure.com/cluster": "MC_aks-rg-mfilrl_aks-cluster-mfilrl_westus2",
23          "kubernetes.azure.com/kubelet-identity-client-id": "82382bec-",
24          "kubernetes.azure.com/mode": "system",
25          "kubernetes.azure.com/node-image-version": "AKSUbuntu-1804gen2containerd-2022.08.23",
26          "kubernetes.azure.com/os-sku": "Ubuntu",
27          "kubernetes.azure.com/role": "agent",
28          "kubernetes.azure.com/storageprofile": "managed",
29          "kubernetes.azure.com/storagetype": "Premium_LRS",
30          "kubernetes.io/arch": "amd64",
31          "kubernetes.io/hostname": "aks-nodepool1-145",
32          "kubernetes.io/os": "linux",
33          "kubernetes.io/role": "agent",
34          "node-role.kubernetes.io/agent": "",
35          "node.kubernetes.io/instance-type": "Standard_DS2_v2",
36          "storageprofile": "managed",
37          "storagetype": "Premium_LRS",
38          "topology.disk.csi.azure.com/zone": "westus2-1",
39          "topology.kubernetes.io/region": "westus2",
40          "topology.kubernetes.io/zone": "westus2-1"
41        },
42        "annotations": {
```


Note!

Kubetest2 has a predecessor: **Kubetest**, and is still in use in certain test cases.

But the Kubernetes Project now recommends using *Kubetest2* as it is more modular, uses plugin paradigm, and has a simplified code structure.

Try it: <https://github.com/Priyankasaggu11929/kubetest2/tree/step-3/kubetest2-aks>

Source Code: <https://sigs.k8s.io/Kubetest2>

Slides: <https://psaggu.com/assets/osc2023/k2p.pdf>

Contact Information:

#sig-testing, #sig-k8s-infra on slack.k8s.io

Slack: **@psaggu**

Email: priyankasaggu11929@gmail.com

Web: psaggu.com

Try it: <https://github.com/Priyankasaggu11929/kubetest2/tree/step-3/kubetest2-aks>

Source Code: <https://sigs.k8s.io/Kubetest2>

Slides: <https://psaggu.com/assets/osc2023/k2p.pdf>

Contact Information:

#sig-testing, #sig-k8s-infra on slack.k8s.io

Slack: **@psaggu**

Email: priyankasaggu11929@gmail.com

Web: psaggu.com

Questions?