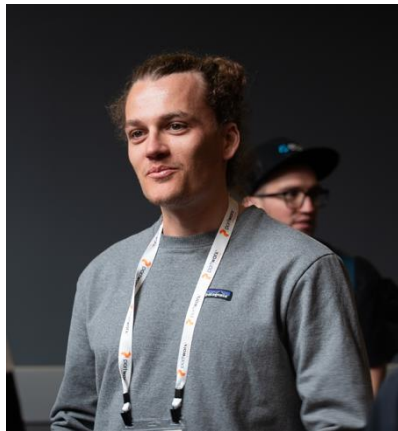# DEFER: THE SILENT HERO OF KUBERNETES OPERATORS

February 11 2026, Fabian Schulz, Lea Brühwiler

Fabian Schulz

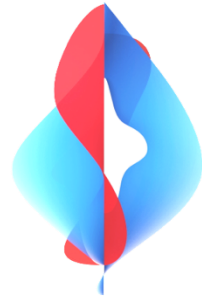DevOps Engineer

fabian.schulz1@swisscom.com

Lea Brühwiler
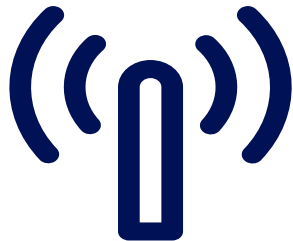
DevOps Engineer

lea.bruehwiler@swisscom.com

User Equipment

(e.g. Phone, IoT Device)

Radio Access Network

Mobile Data Core

Data Network

(e.g. Internet)

Our operators

5G

intent

K8s API

5G core configuration

5G core deployment

Kubernetes from vendor

Network as a service

Networking

routers and switches

OS

Server Hardware

IPAM Inventory

5
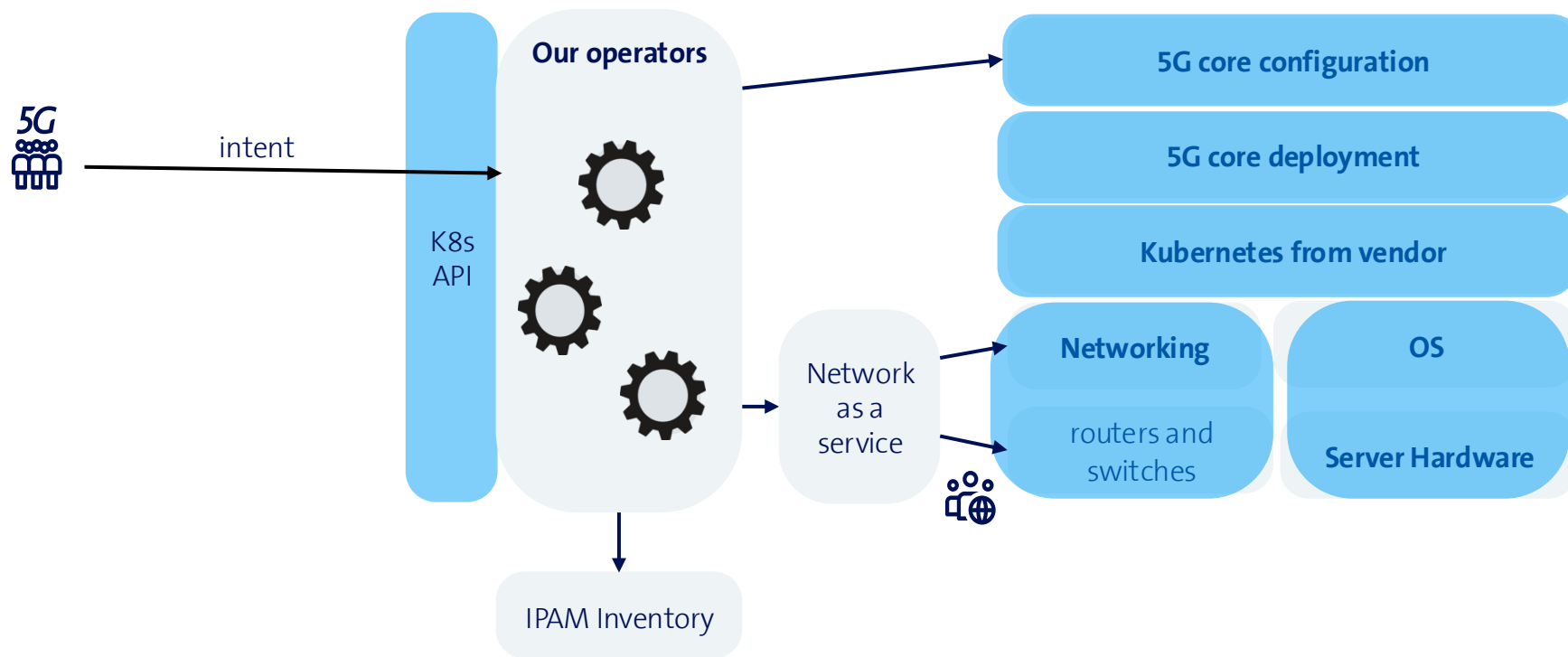
EXAMPLE: HYDROCULTURE OPERATOR

Hydroculture plant

Kubernetes Resource Model - KRM

+

API Extensions

Custom Resource Definitions extend the Kubernetes API.

Custom Resources (CRs) Instances

Custom Resources instantiate a CRD.

Business Logic

Use of Operators or templates to run custom logic.

Herbs CR

kind: Herbs

spec:

  plant: basil

  # temperature: 20°C

  # humidity: 60%

status:

  conditions:

   TemperatureOk: True

   HumidityOk: True

   Ready: True

  generation: 1

  Temperature: 20°C

  Humidity: 60%

{
  Type: TemperatureOk
  Status: True
  ObservedGeneratio: 1
  LastTransitionTime: <timestamp>
  Reason: SelectedTemperatureReached
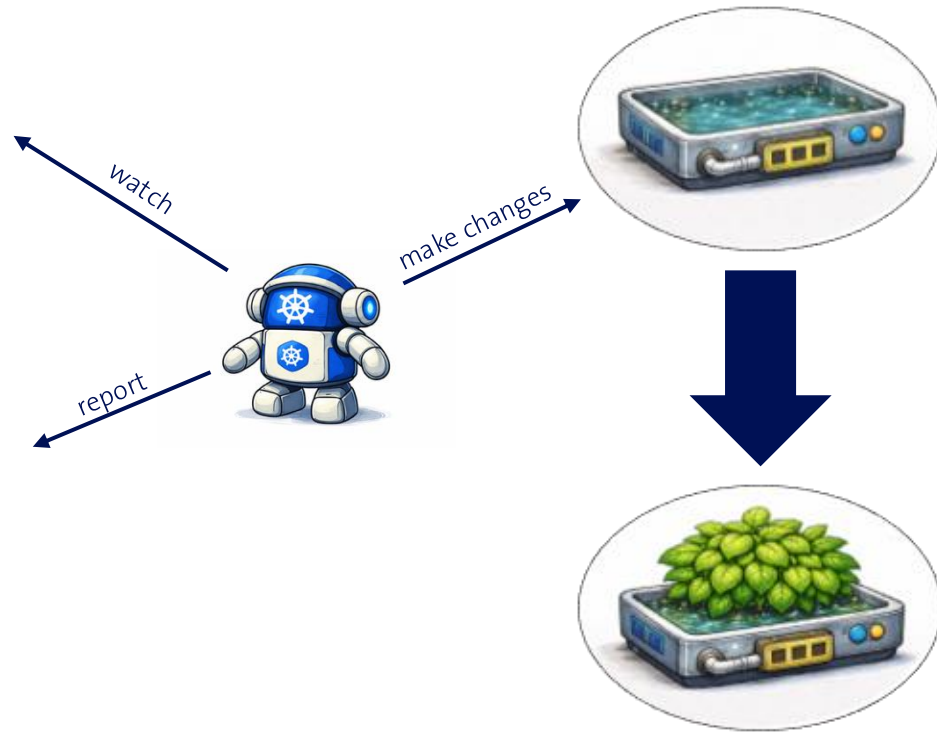 Message: "Temperature of 20°C reached"

kind: Herbs

spec:
 plant: basil

watch

make changes
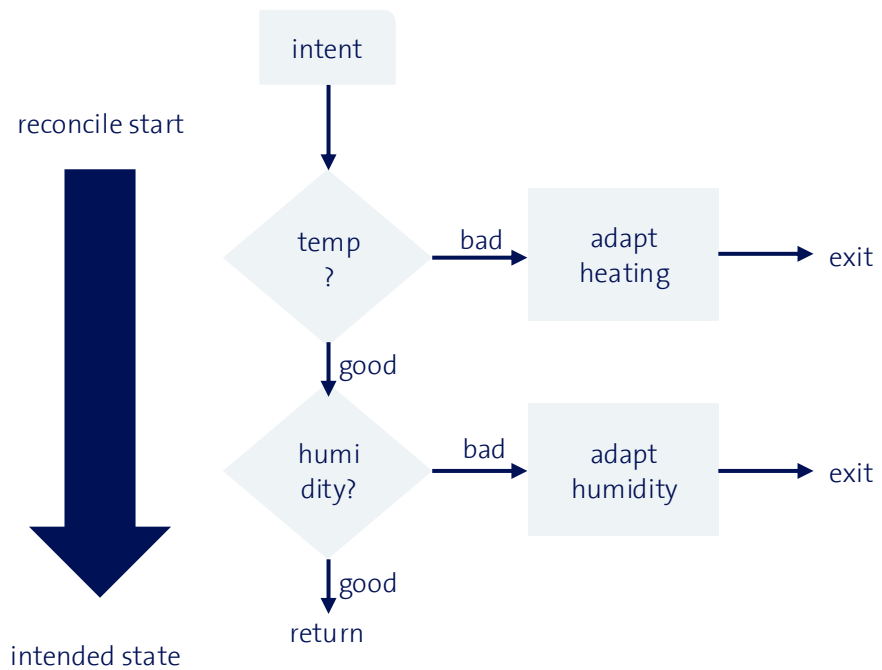
report

status:
 conditions:
  TemperatureOk: True
  HumidityOk: True
  Ready: True
 generation: 1
 Temperature: 20°C
 Humidity: 60%

Herbs Reconcile Loop

CUSTOM RESOURCE STATUS UPDATES

Hydroculture Example

reconcile start

temp?

bad → set temp condition to false → adapt heating → exit

good → set temp condition to true

humidity?

bad → set humidity condition to false → adapt humidity → exit

good → set humidity condition to true → set ready condition to true → exit

intended state

13

temp? → bad → set temp condition to false → set ready condition to false → adapt heating → exit

temp? ↓ good

set temp condition to true ↓

humidity? → bad → set humidity condition to false → set ready condition to false → adapt humidity → exit

humidity? ↓ good

set humidity condition to true ↓

fertilizer? → bad → set fertilizer condition to false → set ready condition to false → adapt fertilizer → exit

fertilizer? ↓ good

set fertilizer condition to true → set ready condition to → exit
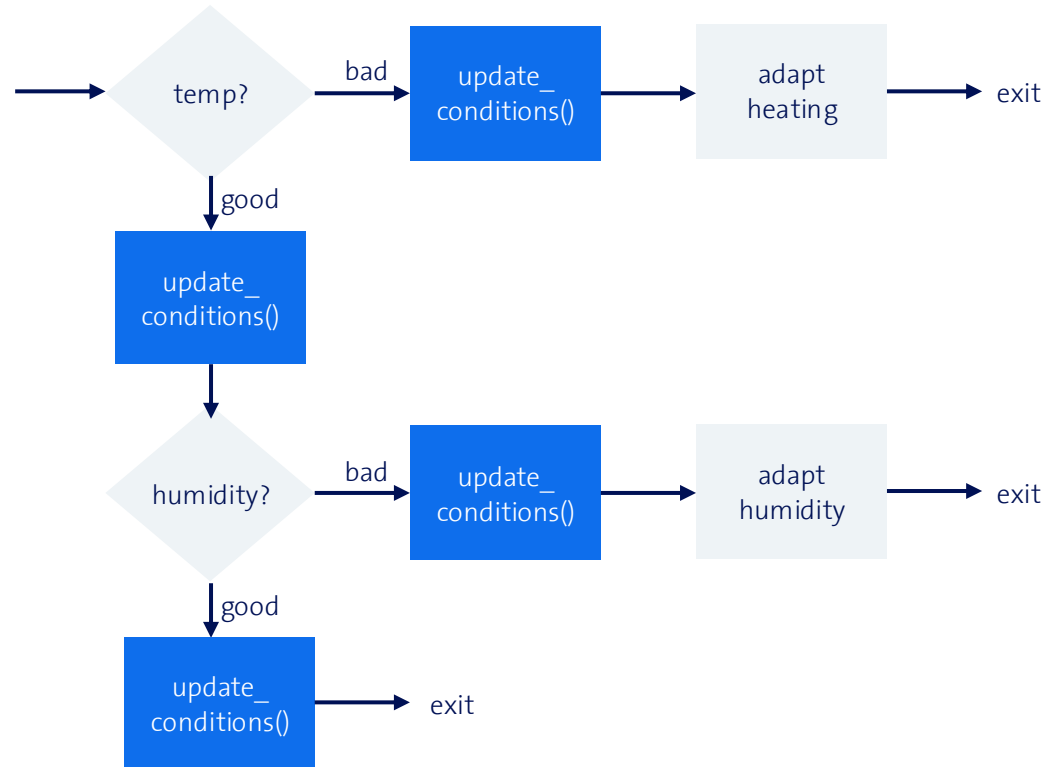
19

Add web cams for the most important plants.

Observation

- Status conditions are updated in many locations in the code
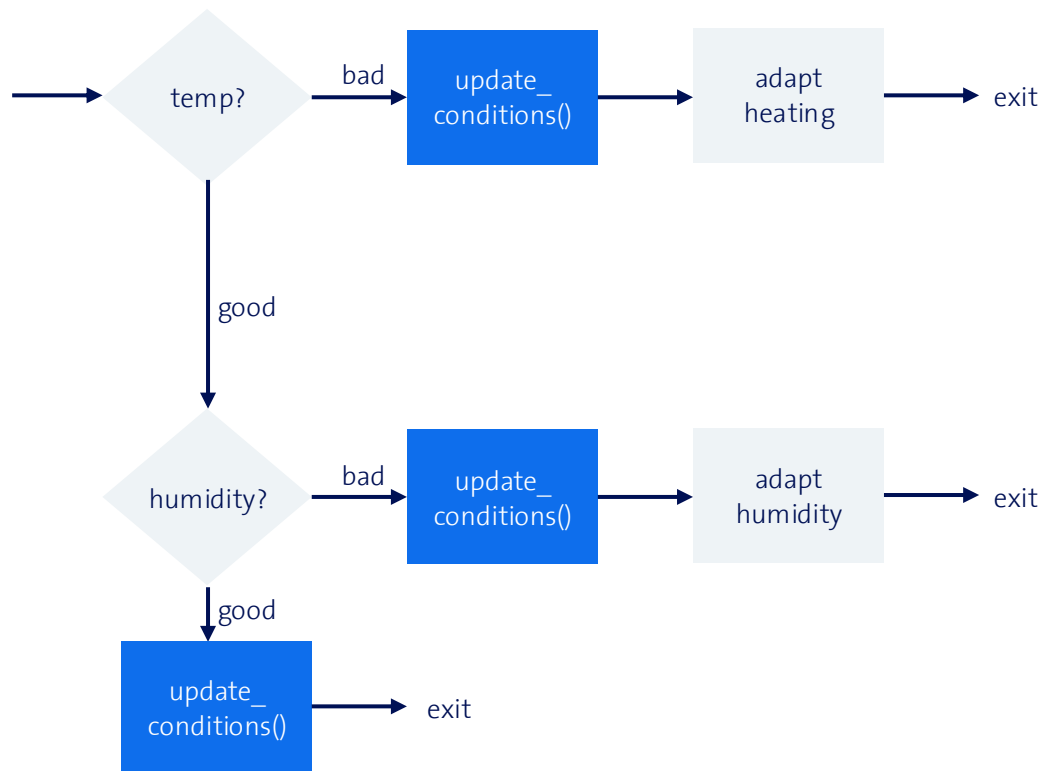- Can easily be forgotten

Solution to remove duplicates -> def update_conditions()

Solution to remove duplicates -> def update_conditions()

defer()

defer()

Defer is used to ensure that a function call is performed later in a program's execution, usually for purposes of cleanup.

What will be returned by the function on the right?

Code snippet

```
func quote() {
  fmt.Println("decide")
  defer fmt.Println("you must")
  fmt.Println("your path")
  }
  return
}
```

Answer

decide

your path

you must
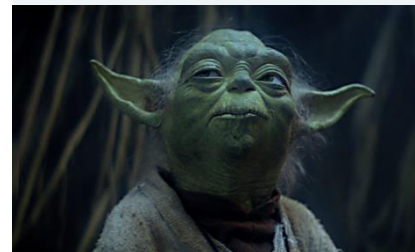
What will be returned by the function on the right?

Code snippet

```go
func number() {
  i := 0
  defer fmt.Println(i)
  i++
}
```

Answer

"0"

What will be returned by the function on the right?

Code snippet

```
func counter() {
  for i := 0; i < 4; i++ {
    defer fmt.Print(i)
  }
}
```

Answer

"3210"

32

Better solution to remove duplicates -> defer func()

defer update_conditions()

def update_conditions()

temp?

bad → adapt heating → exit

good

humidity?

bad → adapt humidity → exit

good

exit

33
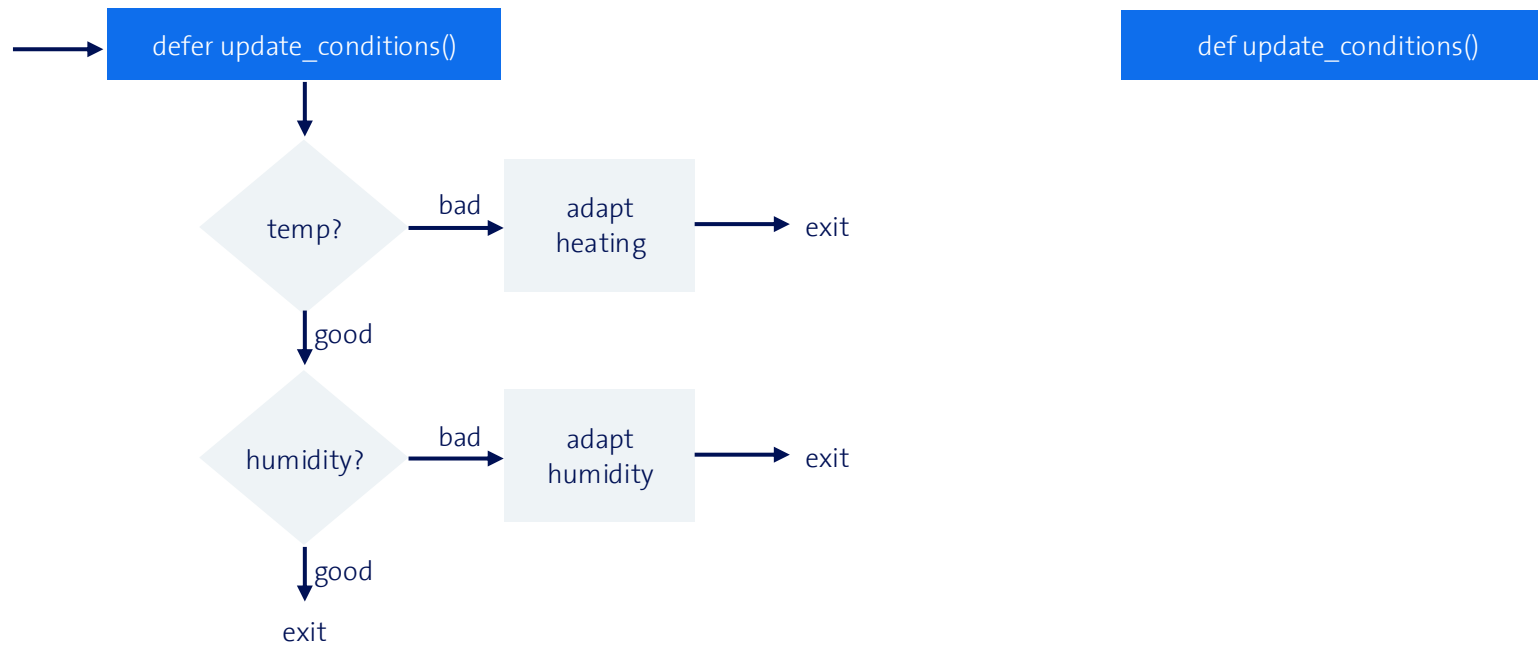
```go
func (r *HerbsReconciler) Reconcile(
        ctx context.Context, req ctrl.Request)
        (ctrl.Result, error) {

        herbs := &hydroculturev1.Herbs{}
        r.Get(ctx, req.NamespacedName, herbs)

        defer func() {
                r.update_conditions(ctx, herbs)
        }()

        // ... reconciler logic here ...

        return
}
```

ERROR HANDLING

Error handling

Events

Create an event for the reconciled resource in case of an error

Status conditions

Update Status Condition and add error message to message of status condition

Logs

Create Log message in case of an error

Key message move error handling to defer function to ensure consistency

What will be returned by the function on the right?

Code snippet

```go
func c() (i int){
  defer i++ }()
  return 1
}

func main() {

  fmt.Println(c())

}
```

Answer

"2"

41

```go
func (r *HerbsReconciler) Reconcile(ctx context.Context,
                reqctrl.Request) (result ctrl.Result, reconcileErr error) {
                // Fetch the Herbs instance
                herbs := &hydroculturev1.herbs{}
                r.Get(ctx, req.NamespacedName, herbs); err != nil

                defer func() {
                                // Set Condition and handle error
                                result, reconcileErr = r.finalizeReconciliation(ctx,
                                                herbs,
                                                reconcileErr)
                }()

                // ... reconciler logic here ...
}
```
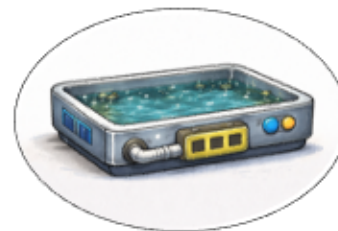
Types of errors

System error

Unexpected errors
Example:
Requests to kubeapi
fail

Domain error

Expected errors
Example:
No basil seeds left in
warehouse

```go
func (r *HydroCultureSettingsReconciler) finalizeReconciliation (ctx context.Context
            o hydroCultureSettings, err error) (result ctrl.Result, err error) {
        if err != nil{
                        // SetReadyConditionFalse adds error message to condition message
                        if updateErr := r.SetReadyConditionFalse(o, err); updateErr != nil{
                                    return errors.Join(updateErr, err)
                        }
                        if ignoreDomainErr(err) != nil{
                                    return ctrl.Result{}, err
                        }
                        return ctrl.Result{Reconcile: true}, nil
        }
        if updateErr := r.SetReadyConditionTrue(o); updateErr != nil{
                        return ctrl.Result{}, updateErr
        }
        return ctrl.Result{}, nil
}
```

SCHEDULED RECONCILIATION

swisscom

```
kind:
HydrocultureSettings
spec:
  plant: basil
```

read

reconcile

> **Ensure backend is in sync with intent**

> **Consistent behaviour**

```go
func (r *HydroCultureSettingsReconciler) finalizeReconciliation (ctx context.Context
                o hydroCultureSettings, err error) (result ctrl.Result, err error) {
        if err != nil{
                        // SetReadyConditionFalse adds error message to condition message
                        if updateErr := r.SetReadyConditionFalse(o, err); updateErr != nil{
                                        return errors.Join(updateErr, err)
                        }
                        if ignoreDomainErr(err) != nil{
                                        return ctrl.Result{}, err
                        }
                        return ctrl.Result{Reconcile: true}, nil
        }
        if updateErr := r.SetReadyConditionTrue(o); updateErr != nil{
                        return ctrl.Result{}, updateErr
        }
        return calculateDurationUntilNextReconciliation(), nil
}
```

RELEASING LOCKED RESOURCES

reconcile start

intended state

49

Example releasing locked resources

```go
func (r *HydroCultureSettingsReconciler) finalizeReconciliation (ctx context.Context
                o hydroCultureSettings, err error) (result ctrl.Result, err error) {
                if errRelease = releaseLock(); errRelease != nil {
                                // Do not return here because the status should still be set
                                err = errors.Join(errRelease, err)
                }
                // ... Logic to update status and calculate next reconciliation here ...

}
```

Takeaways

Use defer in the reconcile function
Separation of concerns improves maintainability and consistency for error handling

Get inspiration from existing projects
Kubernetes Community Guidelines
Flux CD

Ship early
Ship early to gather user feedback

Add extensive testing
Trust in your code changes

Hydroculture demo

Use defer in your Kubernetes controllers!

What will be returned by the function on the right?

Code snippet

```go
func spicy() (x int) {

        x = 1

        defer func() { x *= 10 }()

        defer func() { x += 2 }()

        return x

}
func main() {

        fmt.Println(spicy())

}
```

Answer

"30"

54