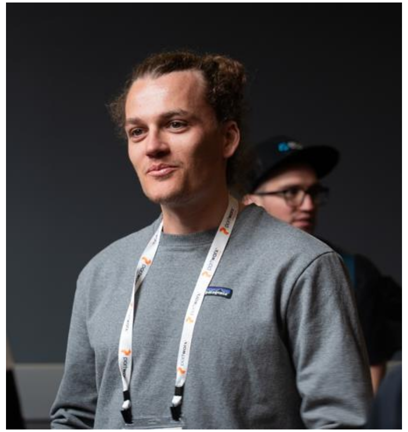
swisscom

DEFER: THE SILENT HERO OF KUBERNETES OPERATORS

February 11 2026, Fabian Schulz, Lea Brühwiler



Fabian Schulz

DevOps Engineer

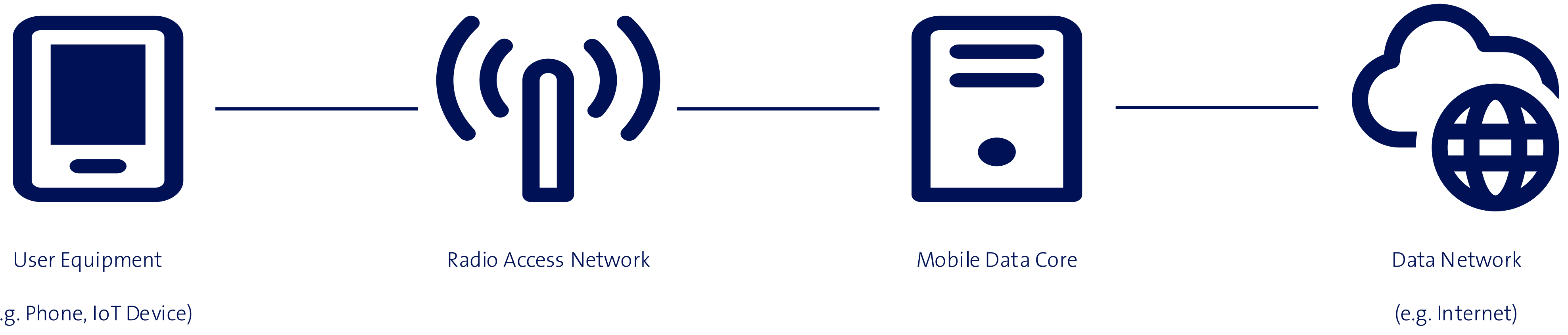
fabian.schulz1@swisscom.com

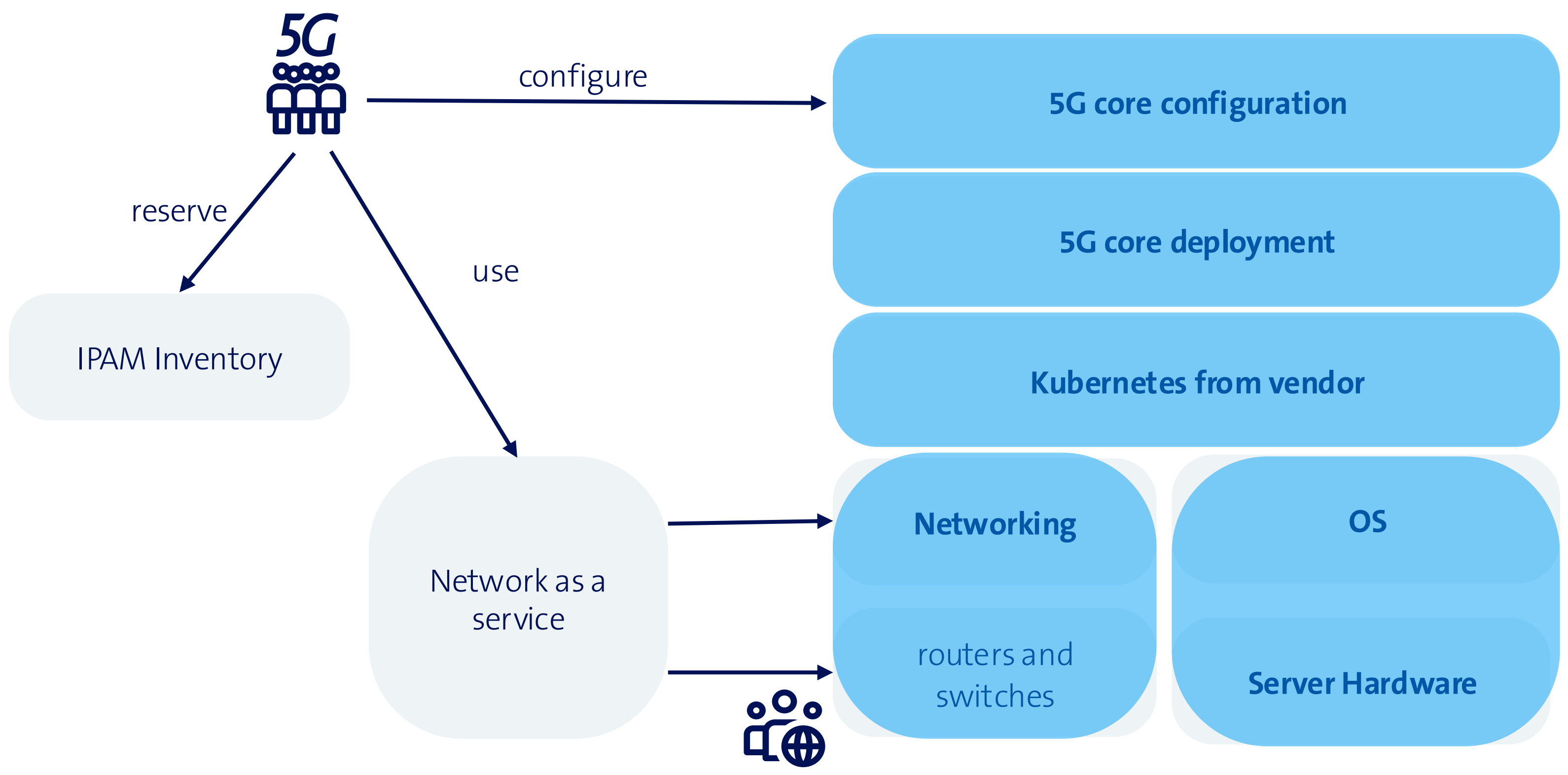
Lea Brühwiler

DevOps Engineer

lea.bruehwiler@swisscom.com

swisscom

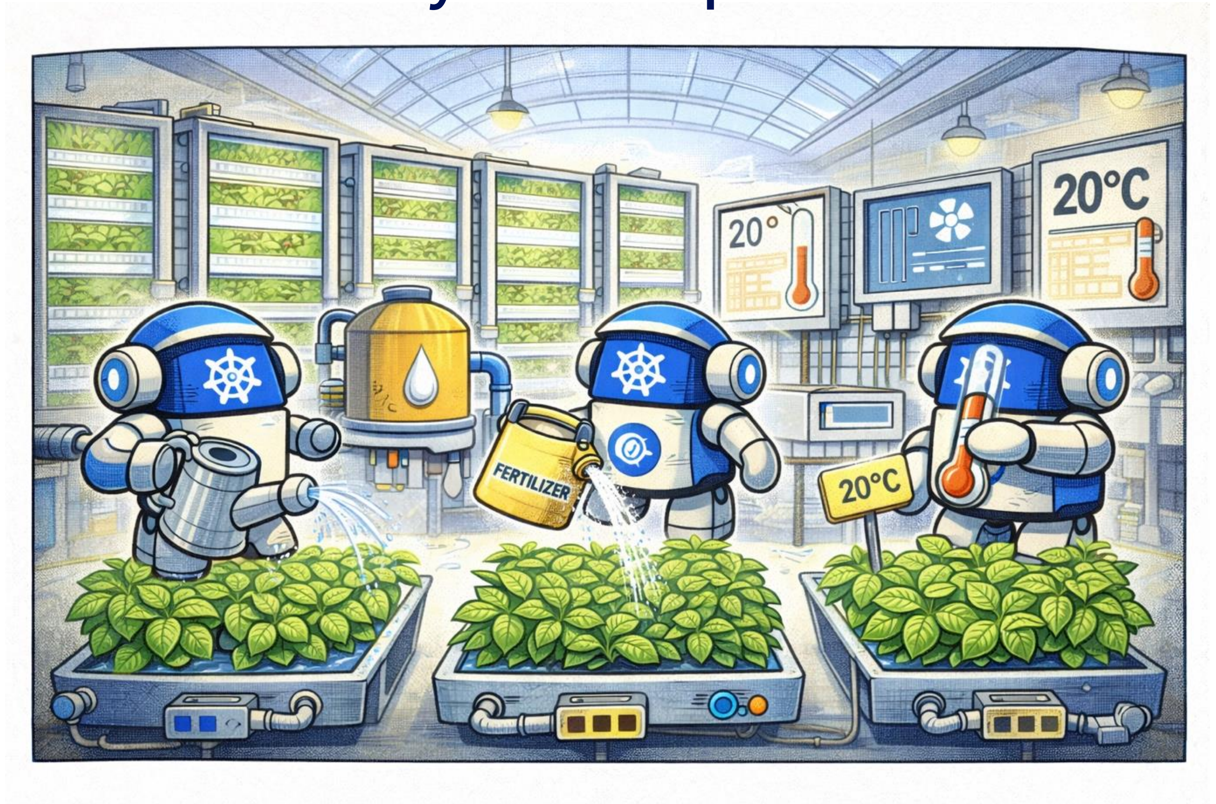

User Equipment
(e.g. Phone, IoT Device)
Radio Access Network
Mobile Data Core
Data Network
(e.g. Internet)


5G
configure
reserve
use
IPAM Inventory
Network as a service
5G core configuration
5G core deployment
Kubernetes from vendor
Networking
routers and switches
OS
Server Hardware

swisscom

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.

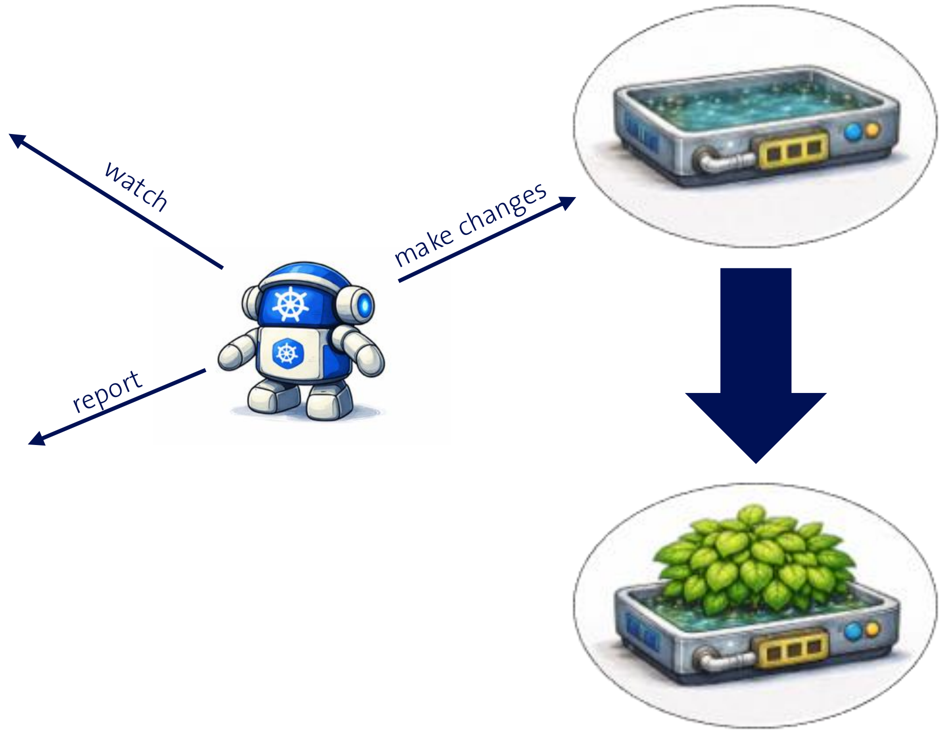
Hydroculture Settings CR

```
kind: HydrocultureSettings
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”
```

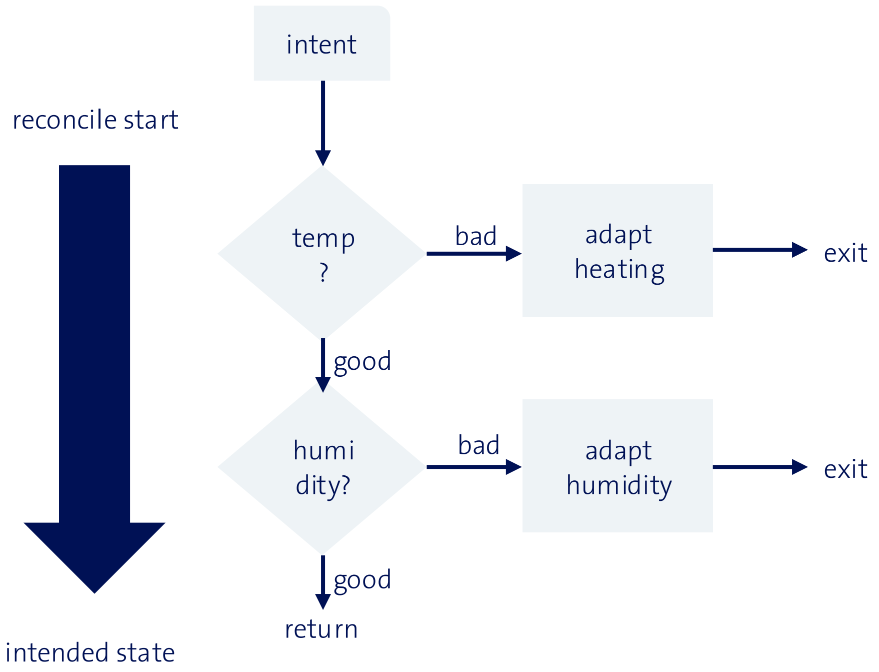
Operator Pattern CNCF Operator WhitePaper

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



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

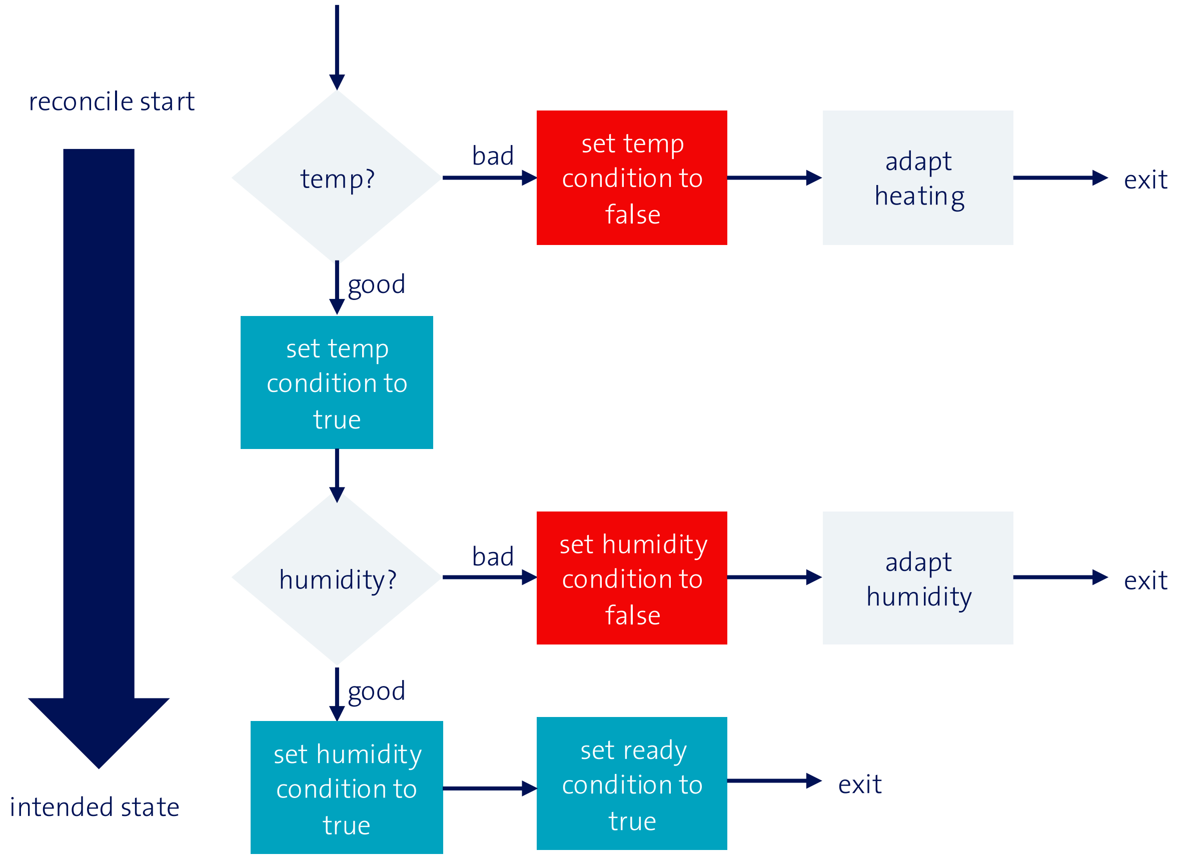
Hydroculture Reconcile Loop

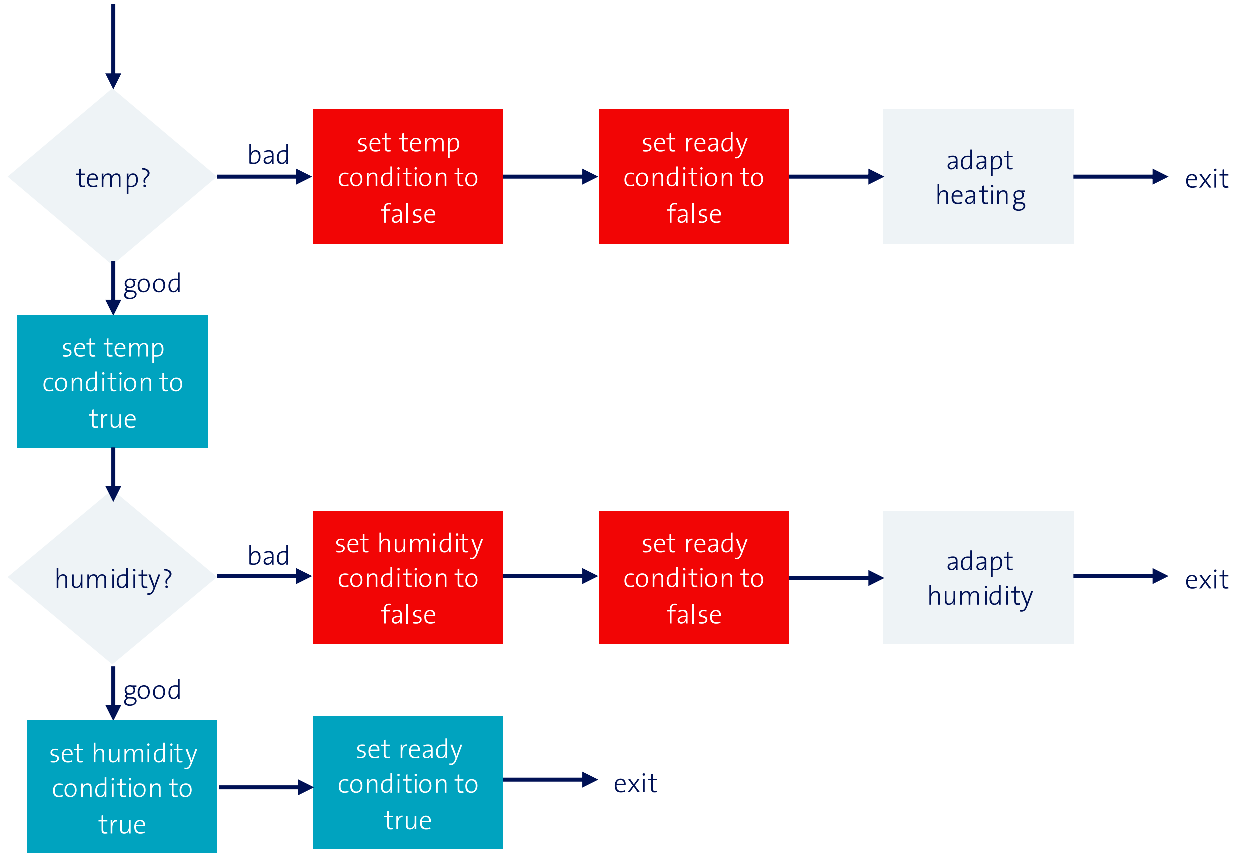


swisscom

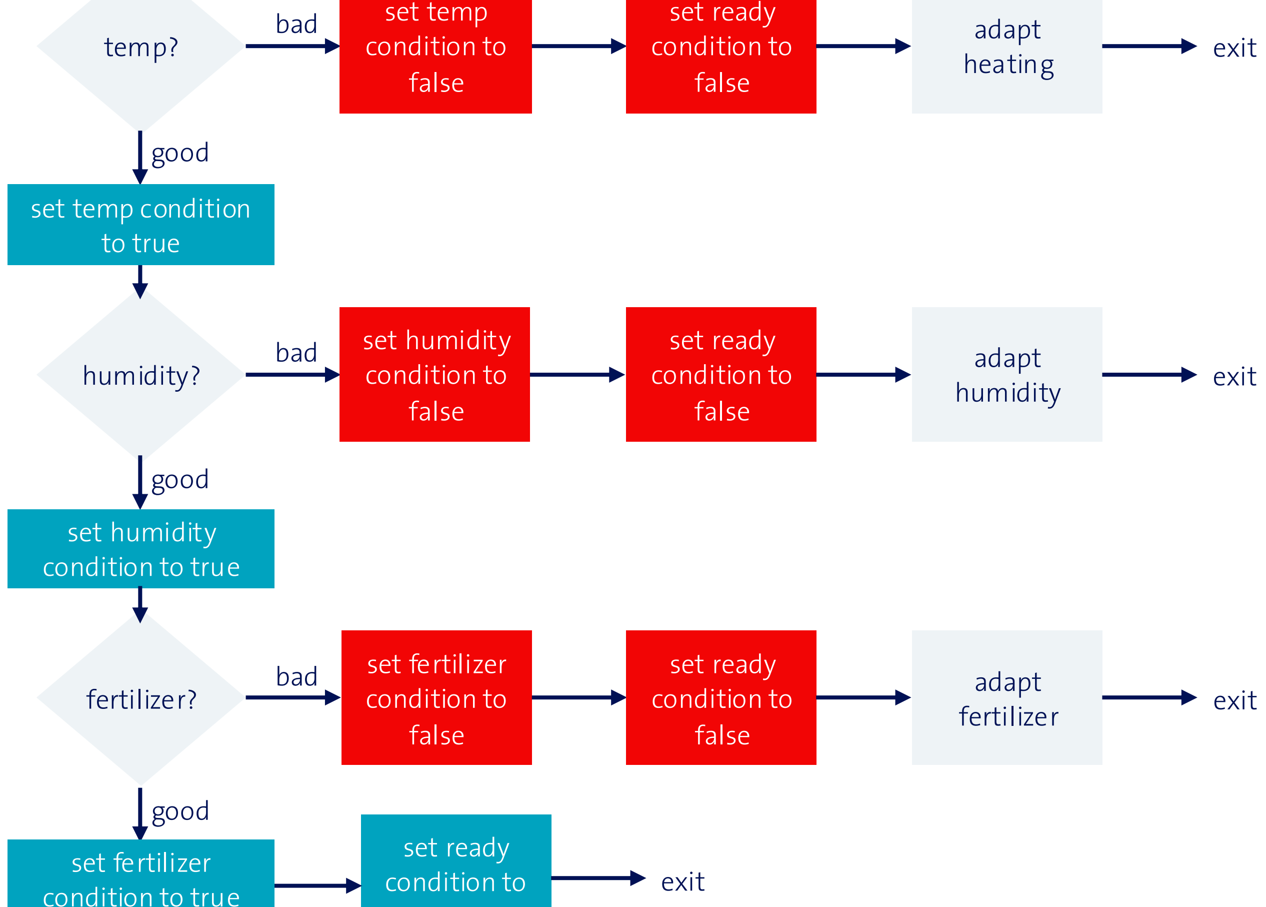
CUSTOM RESOURCE STATUS UPDATES

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

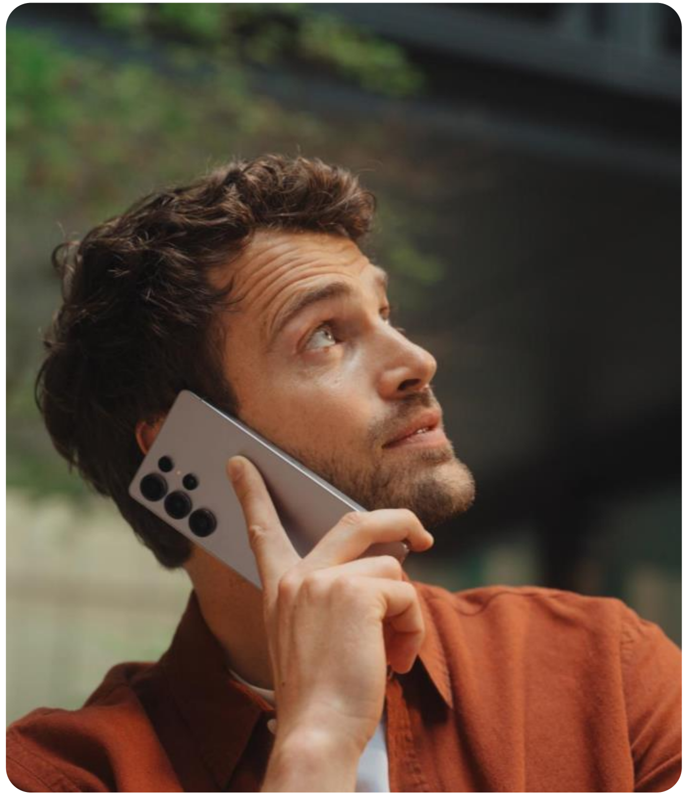

temp?
bad
set temp condition to false
set ready condition to false
adapt heating
exit
good
set temp condition to true
humidity?
bad
set humidity condition to false
set ready condition to false
adapt humidity
exit
good
set humidity condition to true
set ready condition to true
exit

We also need to control the concentration of fertilizer.


temp?
bad
set temp condition to false
set ready condition to false
adapt heating
exit
good
set temp condition to true
humidity?
bad
set humidity condition to false
set ready condition to false
adapt humidity
exit
good
set humidity condition to true
fertilizer?
bad
set fertilizer condition to false
set ready condition to false
adapt fertilizer
exit
good
set fertilizer condition to true
set ready condition to
exit

Add web cams for the most important plants.

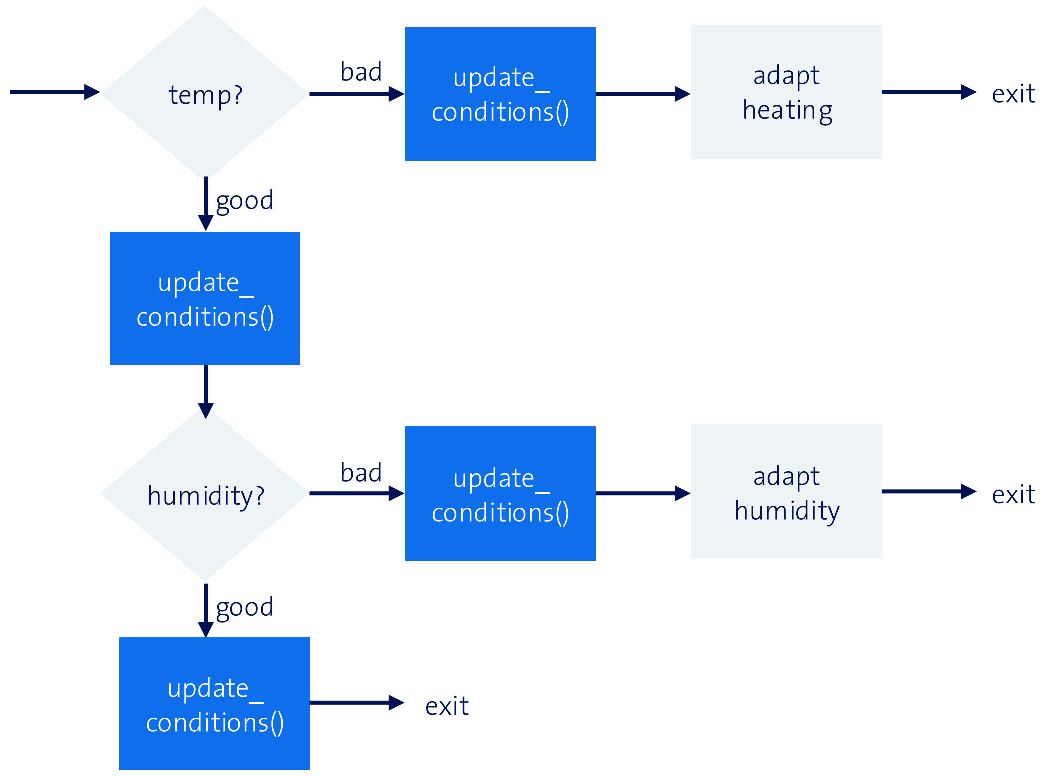
SET STATUS
EVERYWHERE
imgflip.com



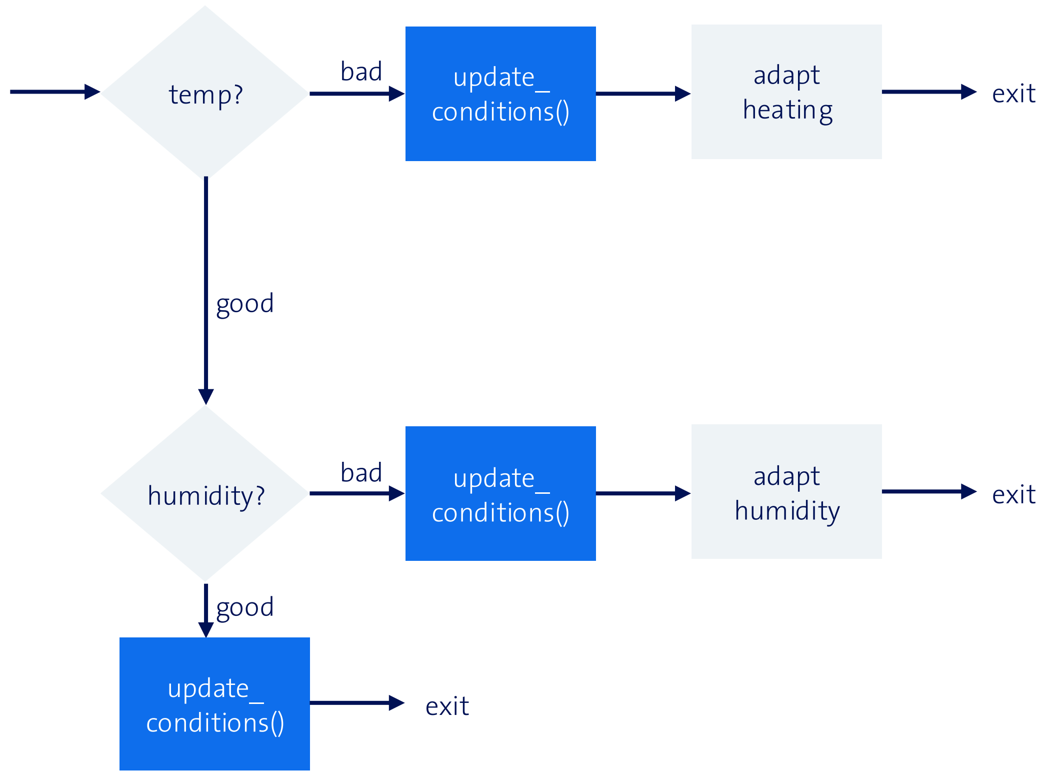
Observation

- 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.

Intro quiz: Question 0

https://go.dev/play/p/eQ882z-pTvU

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

Intro quiz: Question 1

https://go.dev/play/p/BtkBMyt8lHh

What will be returned by the function on the right?

Code snippet

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

Answer

"0"

Intro quiz: Question 2

https://go.dev/play/p/lCDRu_5J0TR

What will be returned by the function on the right?

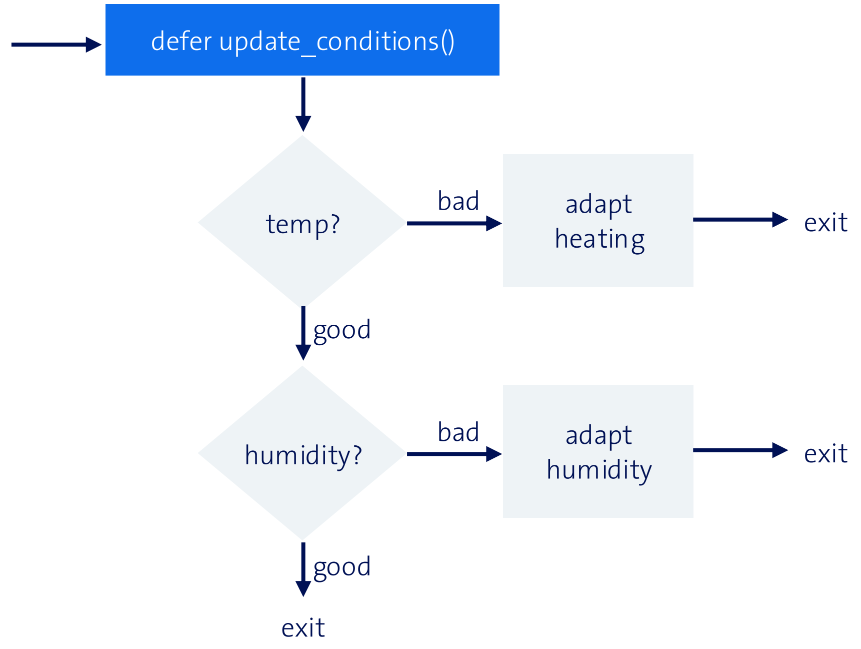
Code snippet

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

Answer

"3210"

Better solution to remove duplicates -> defer func()



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

        hydroCultureSettings := &hydroculturev1.HydroCultureSettings{}
        r.Get(ctx, req.NamespacedName, hydroCultureSettings)

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

        // ... reconciler logic here ...

        return
}
```

swisscom

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

Quiz: Question 3

https://go.dev/play/p/c5BBmkj6DZ-

What will be returned by the function on the right?

Code snippet

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

func main() {

  fmt.Println(c())

}
```

Answer

“2"

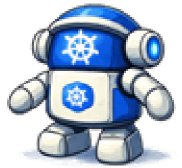
Example error handling

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

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

        // ... reconciler logic here ...
}
```

Types of errors

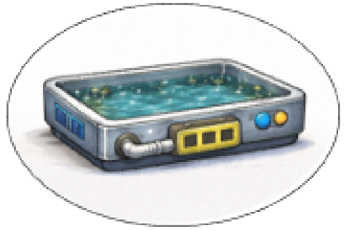
System error

Unexpected errors
Example:
Requests to kubeapi fail



Domain error

Expected errors
Example:
No basil seeds left in warehouse



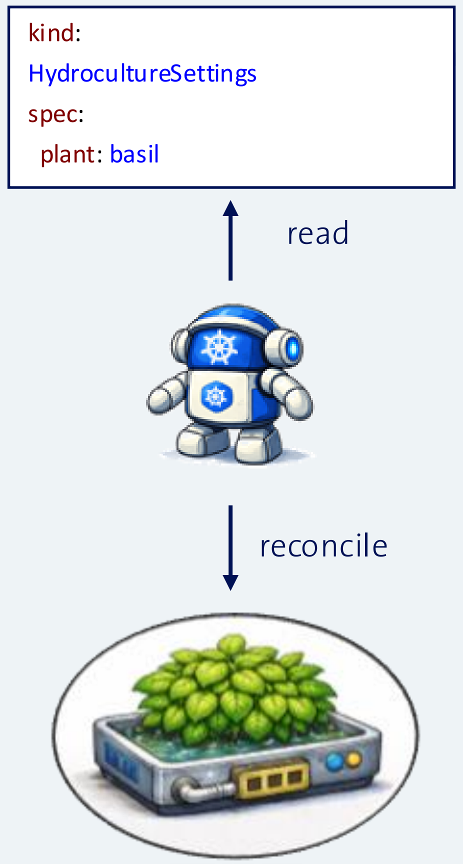
Example handling different error types

```
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
}
```

swisscom

SCHEDULED RECONCILIATION

- Ensure backend is in sync with intent
- Consistent behaviour



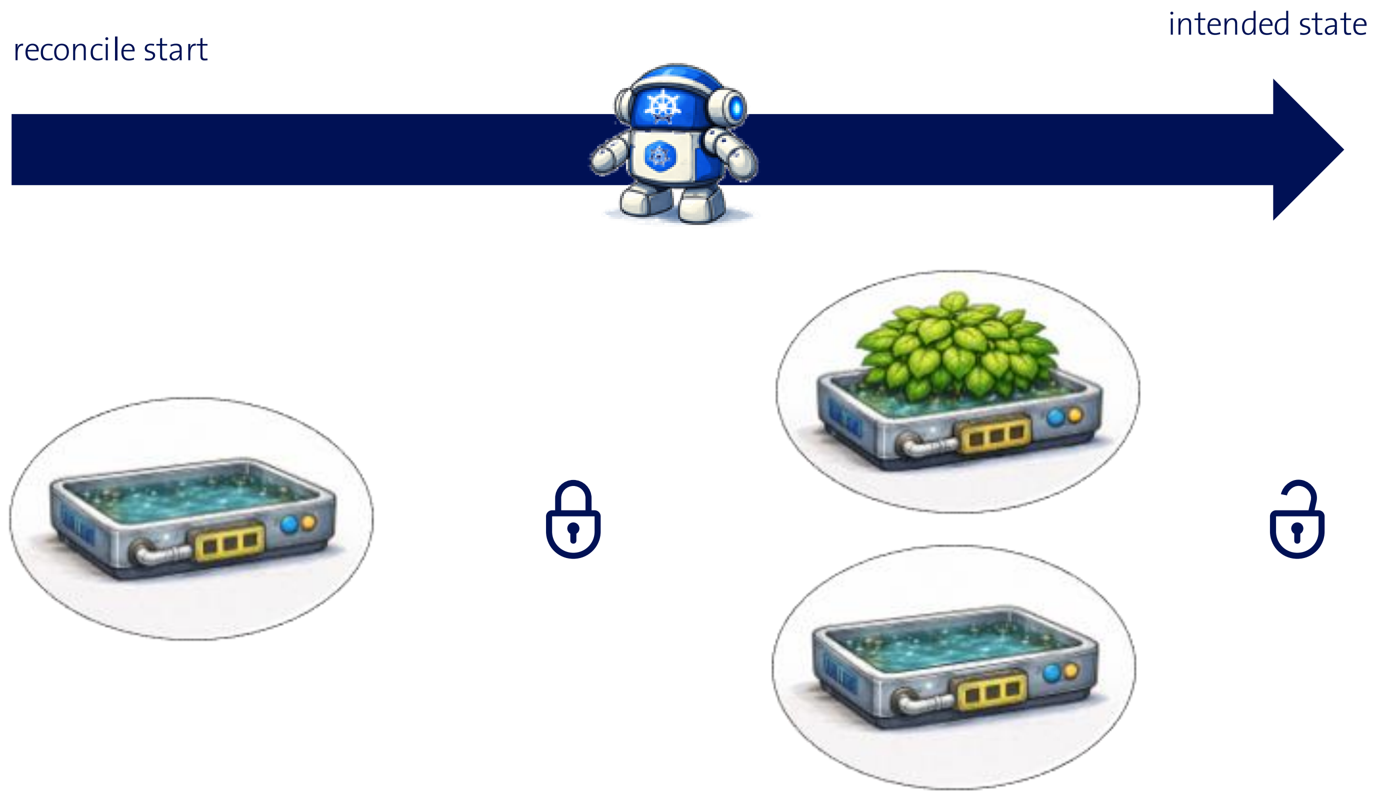
```
kind:
HydrocultureSettings
spec:
  plant: basil
```

Example handling different error types

```
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
}
```

swisscom

RELEASING LOCKED RESOURCES


reconcile start
intended state

Example releasing locked resources

```
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!

Quiz: Question 4

https://go.dev/play/p/Cv2XFvWwUpl

What will be returned by the function on the right?

Code snippet

```
func spicy() (x int) {
        x = 1
        defer func() { x *= 10 }()
        defer func() { x += 2 }()
        return x
}
func main() {
        fmt.Println(spicy())
}
```

Answer

”30"