Yuruna

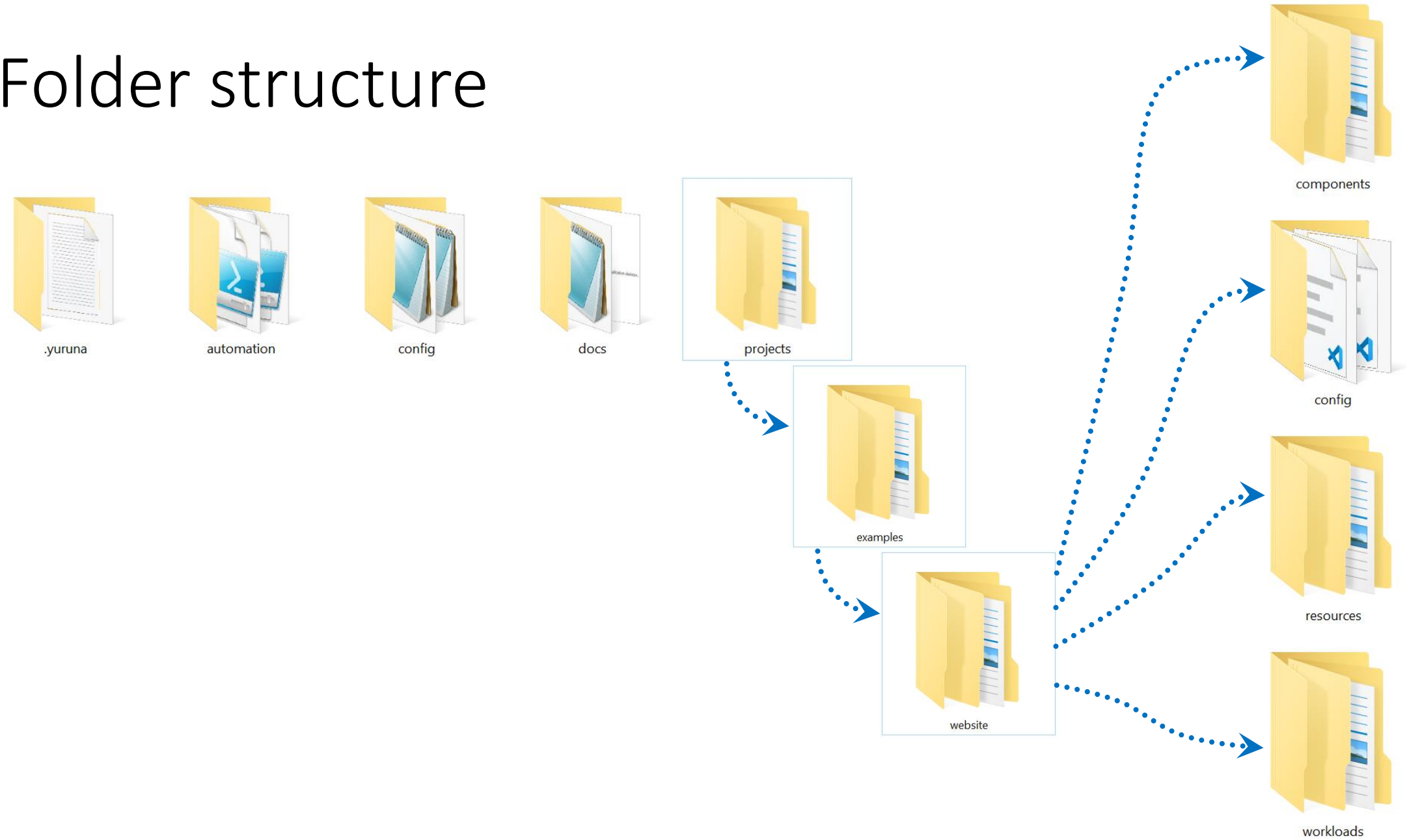
Cross-cloud Kubernetes-based applications

Alisson Sol et al.

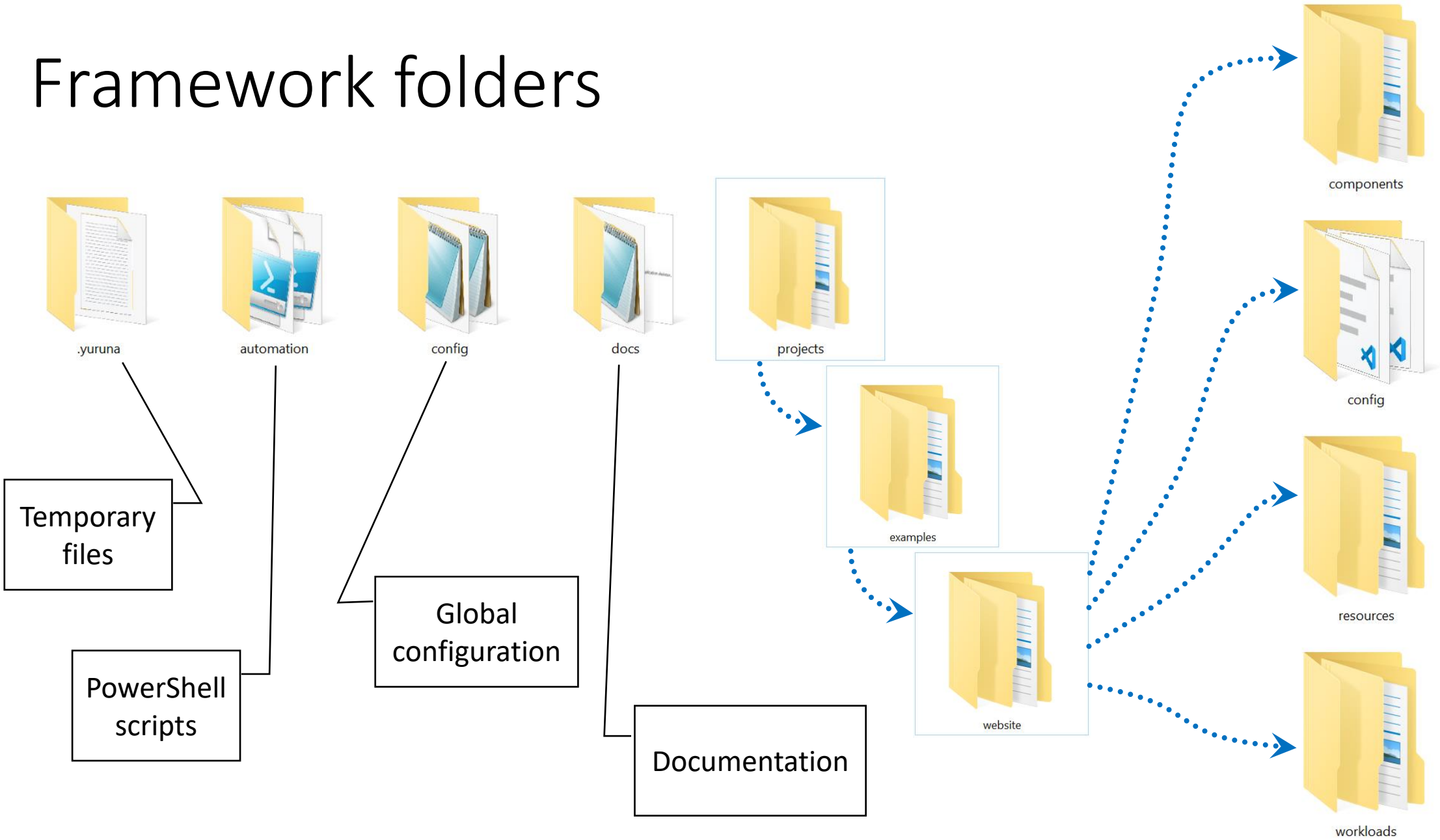
Yuruna goals

- Enable developers to build parameterized K8S applications…
  - … deploying infrastructure resources to clouds in a parameterized way;
  - … building components and pushing to repositories;
  - … installing the workloads in the infrastructure resources.
- Don’t reinvent
  - … deploying infrastructure resources to clouds: uses Terraform
  - … building components: uses Docker containers
  - … deploying workloads: uses Helm

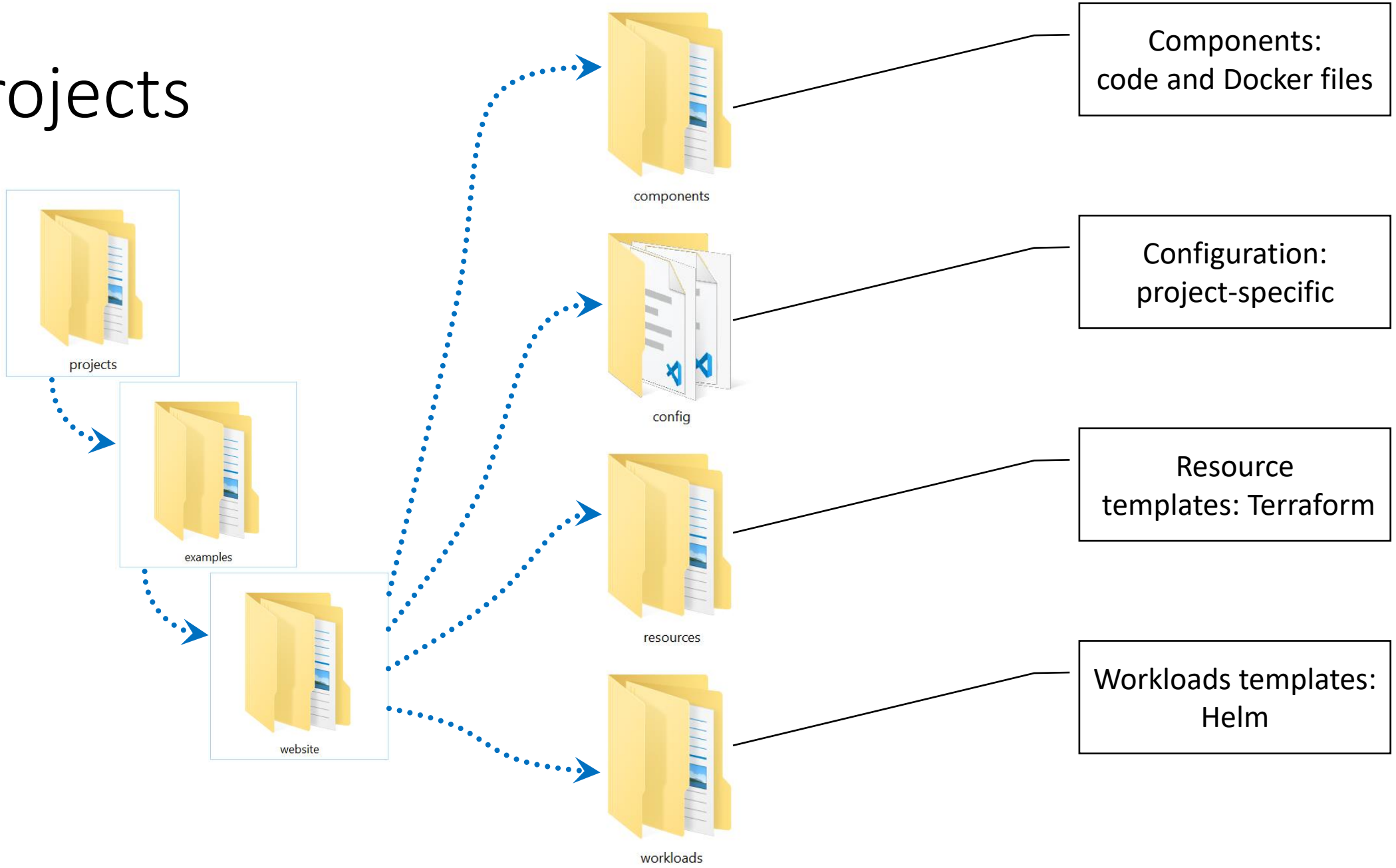
Folder structure



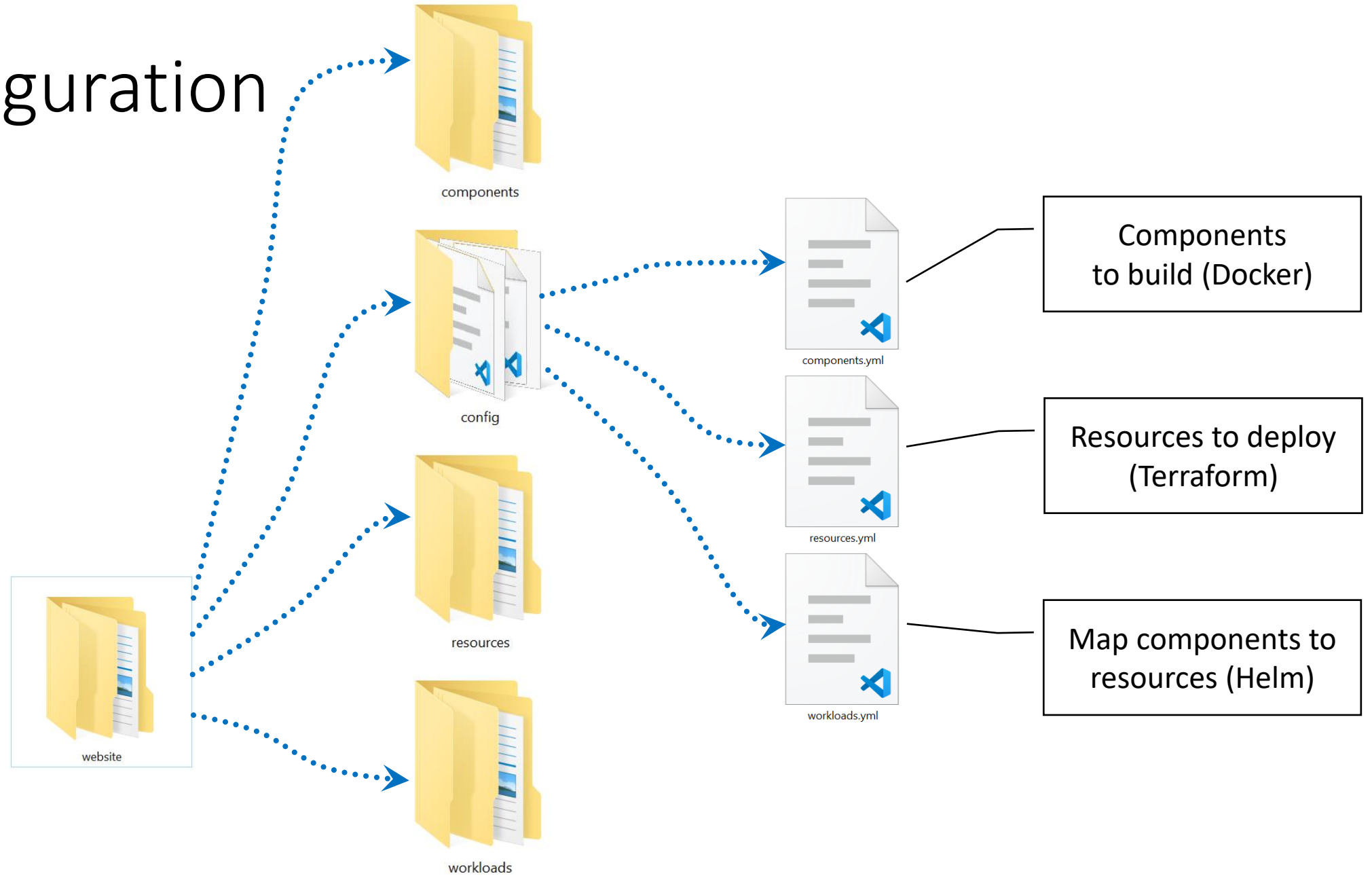
Framework folders



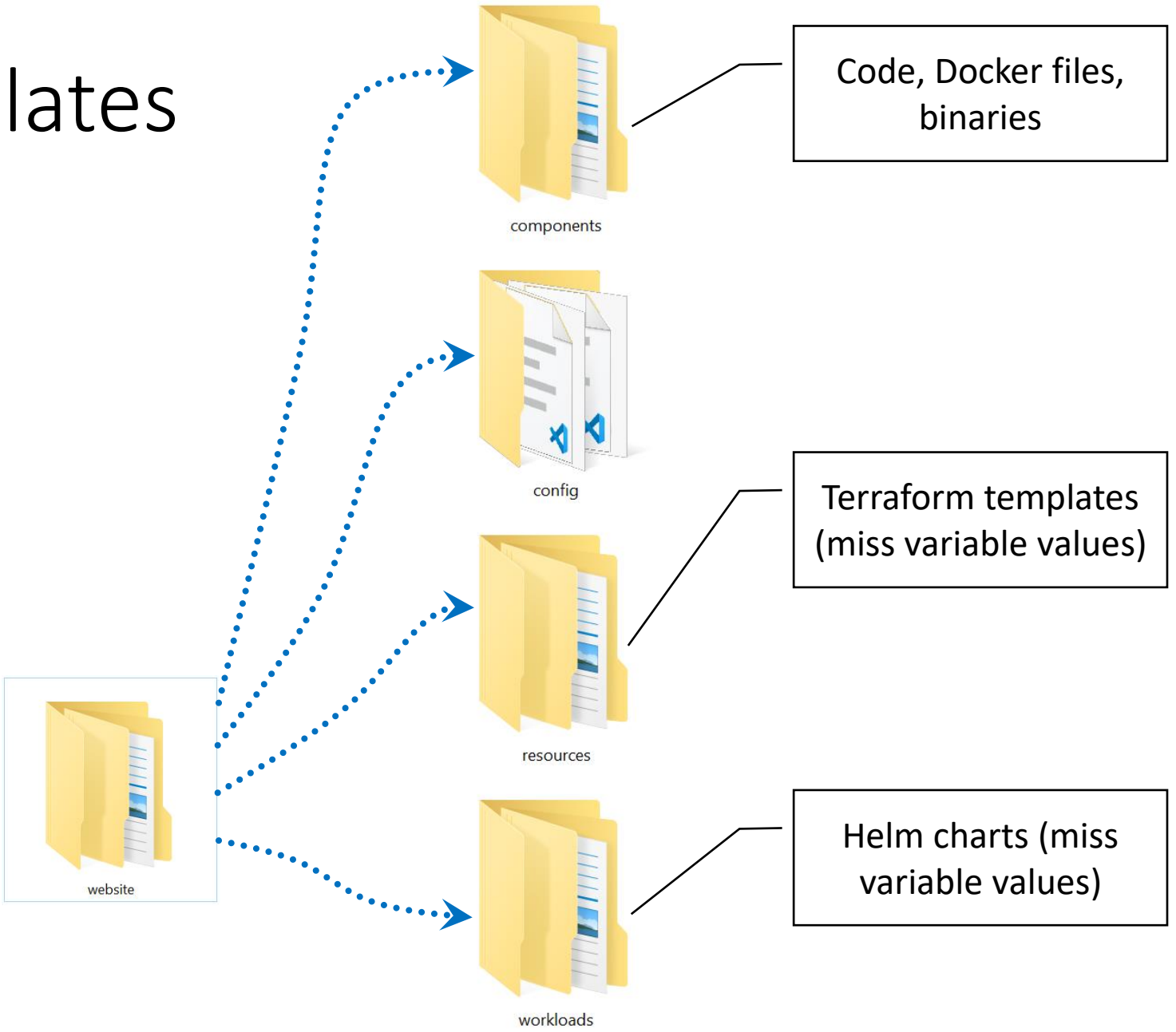
Projects


projects
examples
website
components
config
resources
workloads
Components:
code and Docker files
Configuration:
project-specific
Resource
templates: Terraform
Workloads templates:
Helm

Configuration

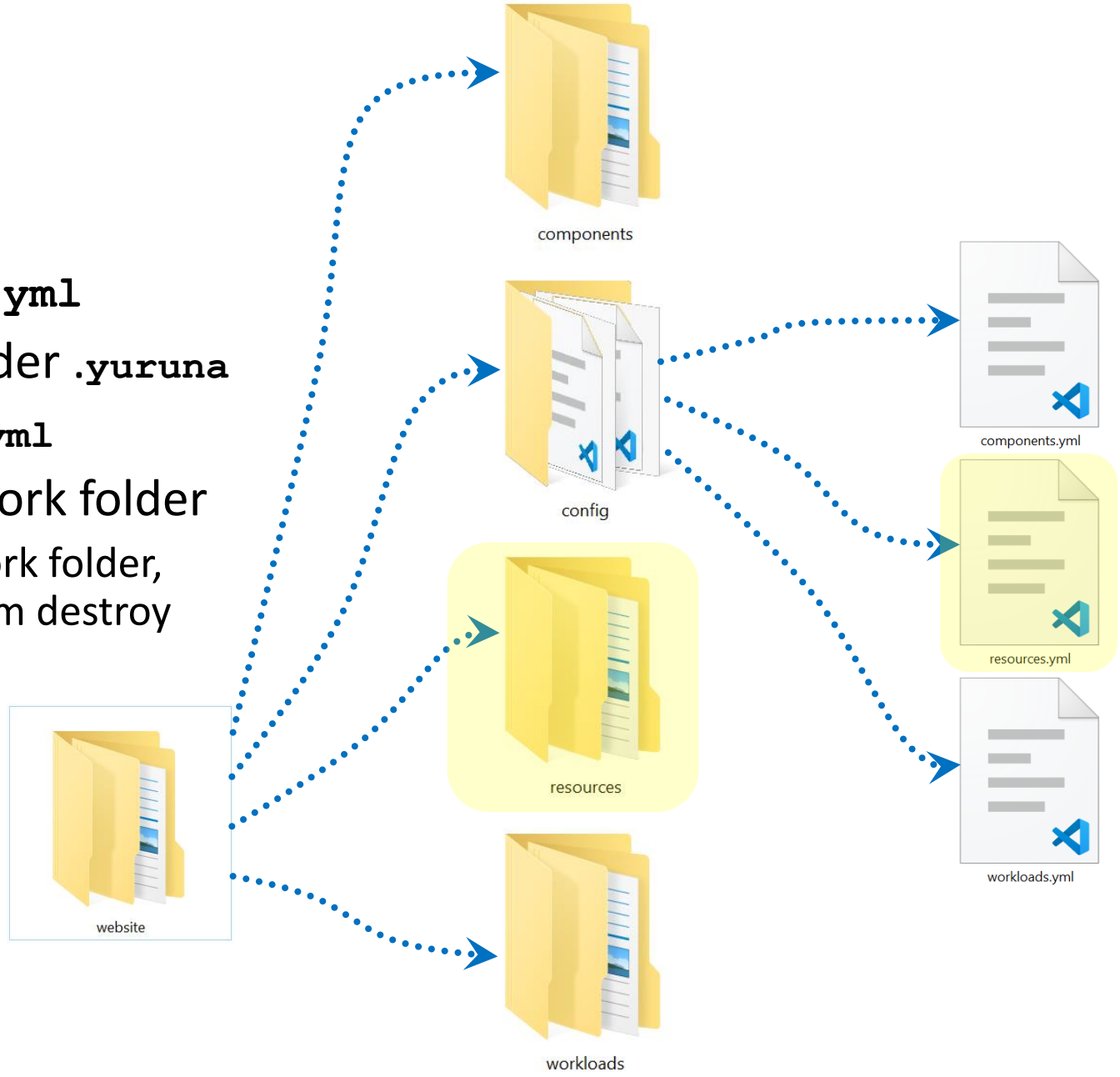


Templates


components
config
resources
workloads
website
Code, Docker files, binaries
Terraform templates (miss variable values)
Helm charts (miss variable values)

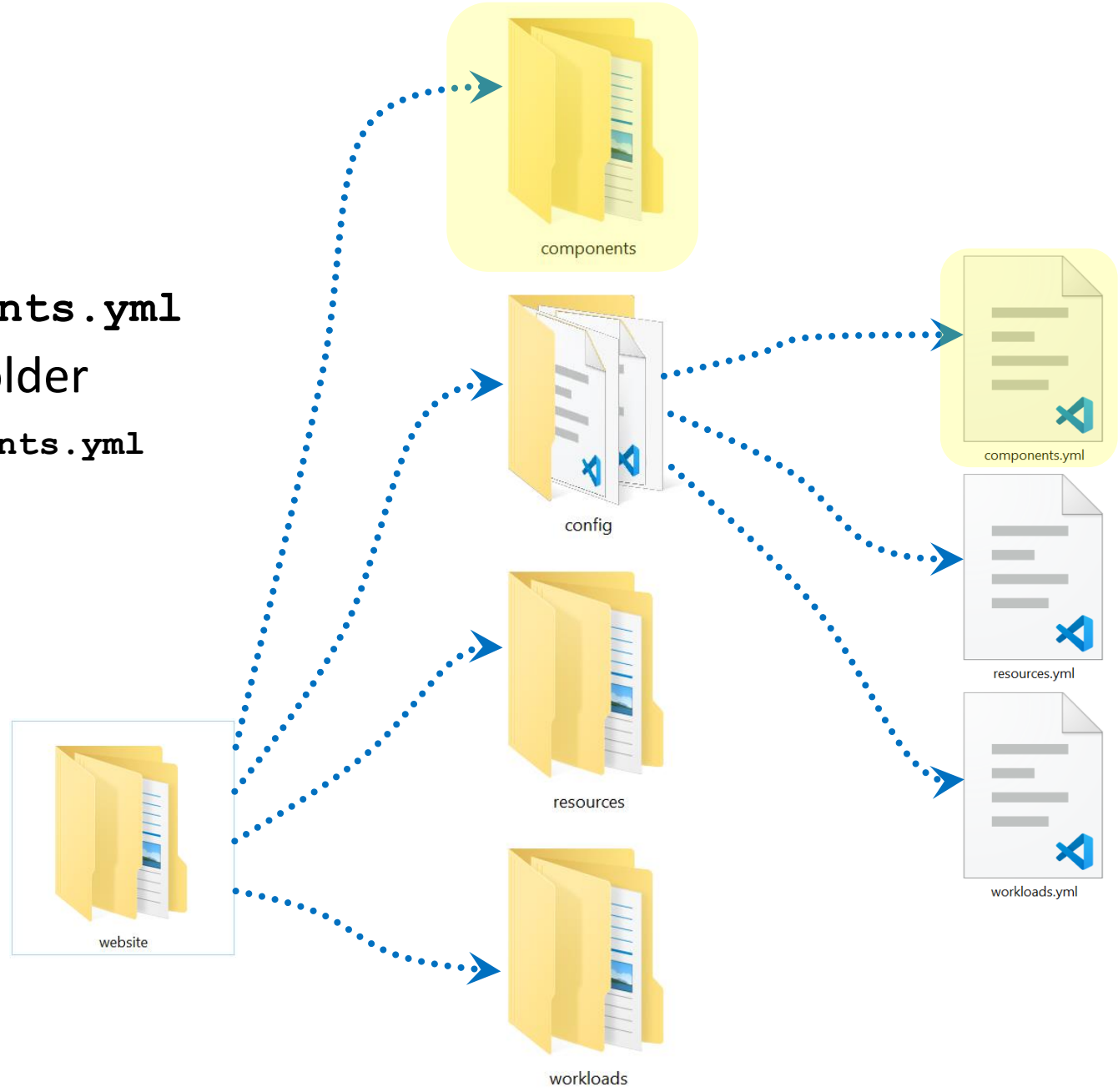
`yuruna resources`

- For each resource in `resources.yml`
  - copy template to work folder under `.yuruna`
  - apply variables from `resources.yml`
  - execute `terraform apply` from work folder
    - creates local `.terraform` under work folder, which can be used later in terraform destroy



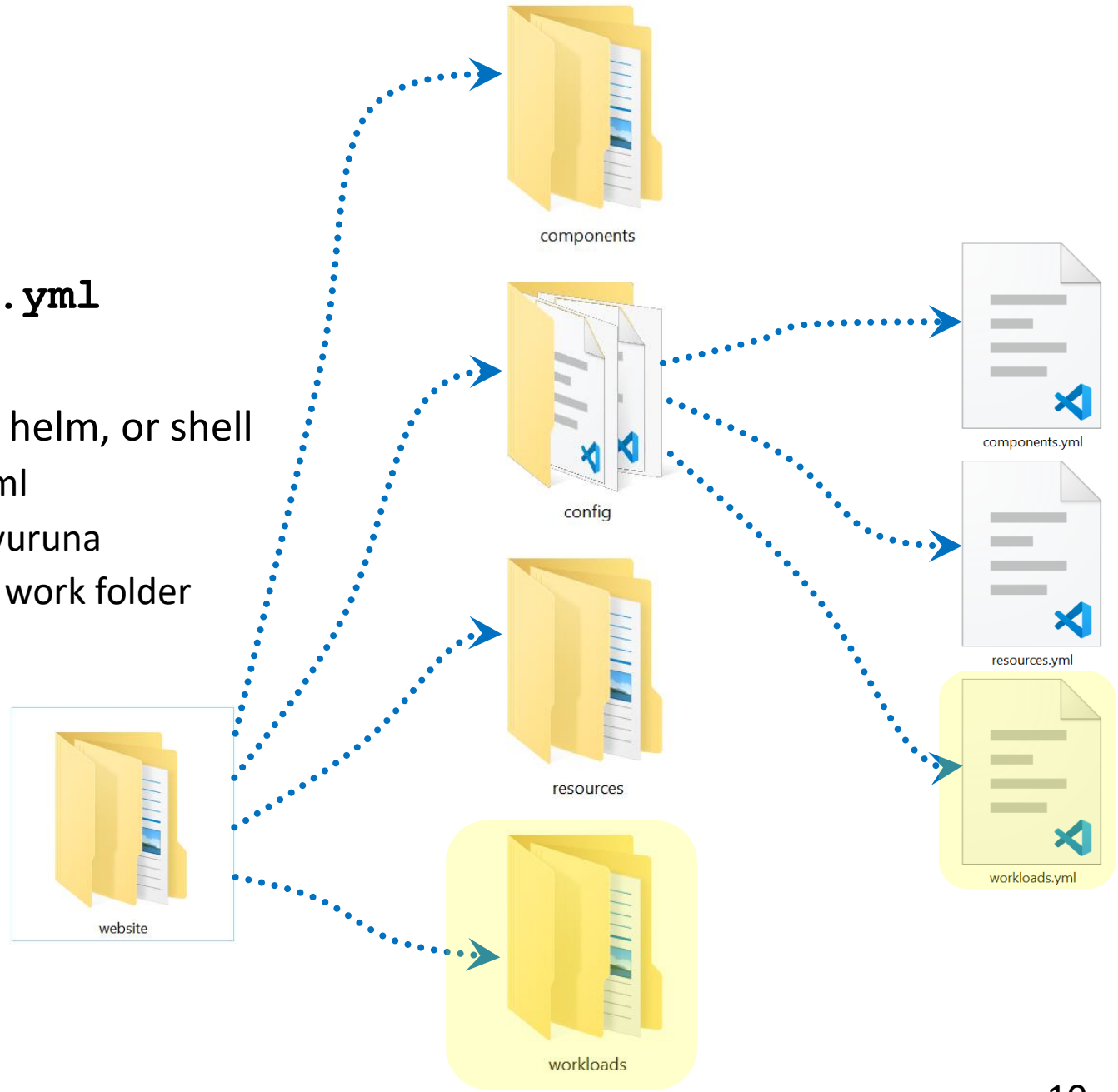
`yuruna components`

- For each component in `components.yml`
  - execute build command in the folder
    - command is parameter in `components.yml`
  - push component to registry



`yuruna workloads`

- For each workload in **`workloads.yml`**
  - switch to context
    - apply deployments: chart, kubectl, helm, or shell
      - apply variables from workloads.yml
      - copy chart to work folder under .yuruna
        - execute **`helm install`** in work folder
      - other expressions use ${env:vars}

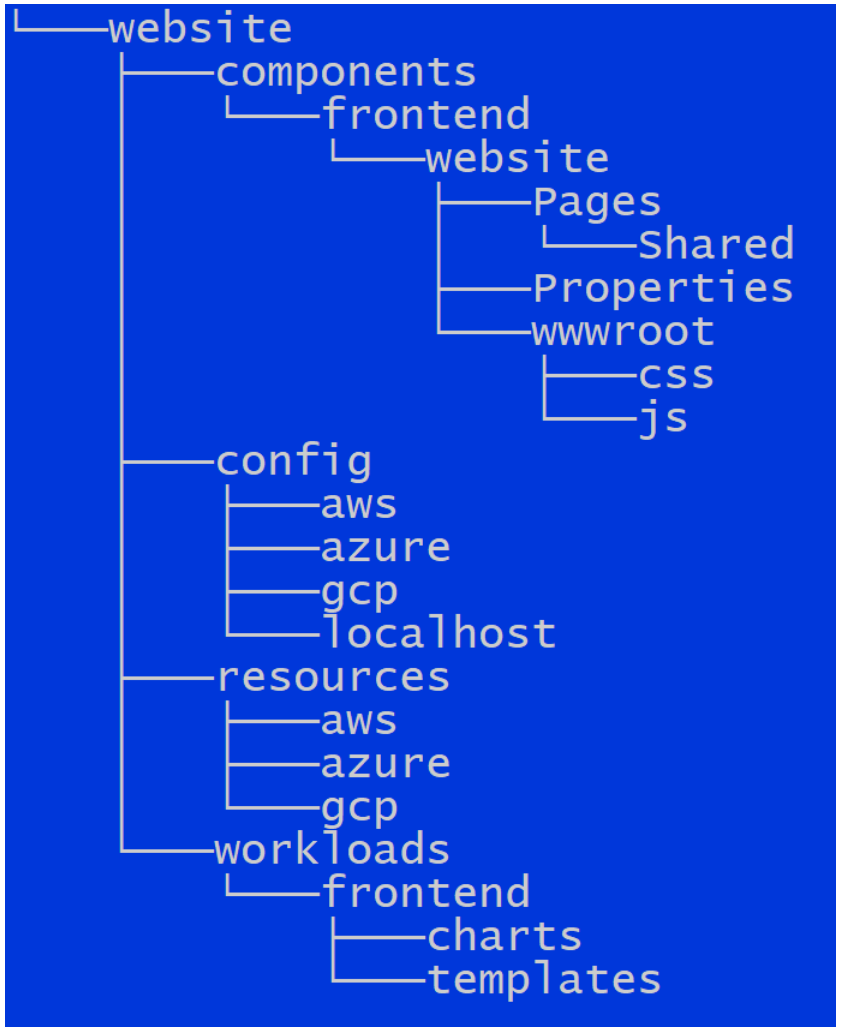


`yuruna` command details

- Creates subfolders under **`.yuruna`** folder with [project] name
  - -name option to “name” workfolder
- There are **`yuruna_root`** and **`project_root`** variables
  - Projects can be somewhere else, away from the **`yuruna_root`**
  - References in files should be relative to **`project_root`**
- Example
  - **`.\yuruna.ps1 workloads C:\Git\yuruna\projects\examples\website localhost`**

Multiple configurations

- Create subfolders under **`config`**
  - Each subfolder needs to have its own copy of the configuration files
    - **`resources.yml`**
    - **`components.yml`**
    - **`workloads.yml`**
  - If such files are found under "**`config`**" then they are used
- Indicate **`config_subfolder`**
  - **`yuruna resources aws`**
  - **`yuruna components azure`**



Resources.yml

```yaml
# Resources information
---
globalVariables:
  namespace: "yuruna"
  clusterDnsPrefix: "yuruna"
  clusterName:      "yuruna"
  clusterRegion:    "westus2"
  clusterVersion:   "1.20.0"
  frontendIpName:   "localhost"
  resourceGroup:    "yuruna"
  resourceTags:     "yuruna"
  nodeCount:        1
  registryName:     "localhost:5000"

resources:
# Localhost with Docker Desktop
# Nothing needs to be created: just provide cluster name
- name:             "docker-desktop"
  template:         ""
  variables:
```

Components.yml

```yaml
# Components information
---
globalVariables:
  containerPrefix:  "yuruna"
  registryLocation: "localhost:5000"
  registryName:     "localhost"
  buildCommand: docker build --rm -f $dockerfile -t "${containerPrefix}/${project}:latest" "$buildPath"
  tagCommand: docker tag "${containerPrefix}/${project}:latest" "${registryLocation}/${containerPrefix}/${project}:latest"
  pushCommand: docker push "${registryLocation}/${containerPrefix}/${project}:latest"

components:
- project: "website"
  buildPath: "frontend/website"
```

Workloads.yml (part 1)

```yaml
# Workloads information
---
globalVariables:
  namespace:        "yuruna"
  containerPrefix:  "yuruna"
  registryLocation: "localhost:5000"
  registryName:     "localhost"
  ipName:           "localhost"
  ipAddres:         "127.0.0.1"
  site:             "localhost"
  certManagerIssuerEmail: "certificates@yuruna.com"
  dockerUsername:   "dummy"
  dockerPassword:   "dummy"
```

Workloads.yml (part 2)

```
workloads:
- context: "docker-desktop"
  deployments:
  - kubectl: "create namespace ${env:namespace}"
  - kubectl: "config set-context --current --namespace=${env:namespace}"
  - kubectl: "delete secret registry-credential"
  - kubectl: "create secret docker-registry registry-credential --docker-server=http://${env:registryLocation} --docker-
username=${env:dockerUsername} --docker-password=${env:dockerPassword}"
  - helm: "repo add ingress-nginx https://kubernetes.github.io/ingress-nginx"
  - helm: "repo update"
  - helm: "uninstall nginx-ingress --namespace ${env:namespace}"
  - helm: >
      install nginx-ingress ingress-nginx/ingress-nginx
      --namespace ${env:namespace}
      --set controller.replicaCount=2
      --set controller.nodeSelector."beta\.kubernetes\.io/os"=linux
      --set defaultBackend.nodeSelector."beta\.kubernetes\.io/os"=linux
      --set controller.service.loadBalancerIP="${env:ipAddres}"
      --set controller.service.annotations."service\.beta\.kubernetes\.io/azure-dns-label-name"="${env:ipName}"
      --set controller.service.annotations."kubernetes\.io/ingress\.global-static-ip-name"="${env:ipName}"
  - kubectl: "apply --validate=false -f https://github.com/jetstack/cert-manager/releases/download/v1.1.0/cert-manager.yaml"
  - chart: "frontend/website"
    variables:
```

Backyard

- CNCF artwork: https://github.com/cncf/artwork
- Yuruna: https://bit.ly/asol-yrn
- PowerShell Best Practices: http://powershell-guru.com/