Yuruna

Cross-cloud Kubernetes-based applications

Alisson Sol et al.

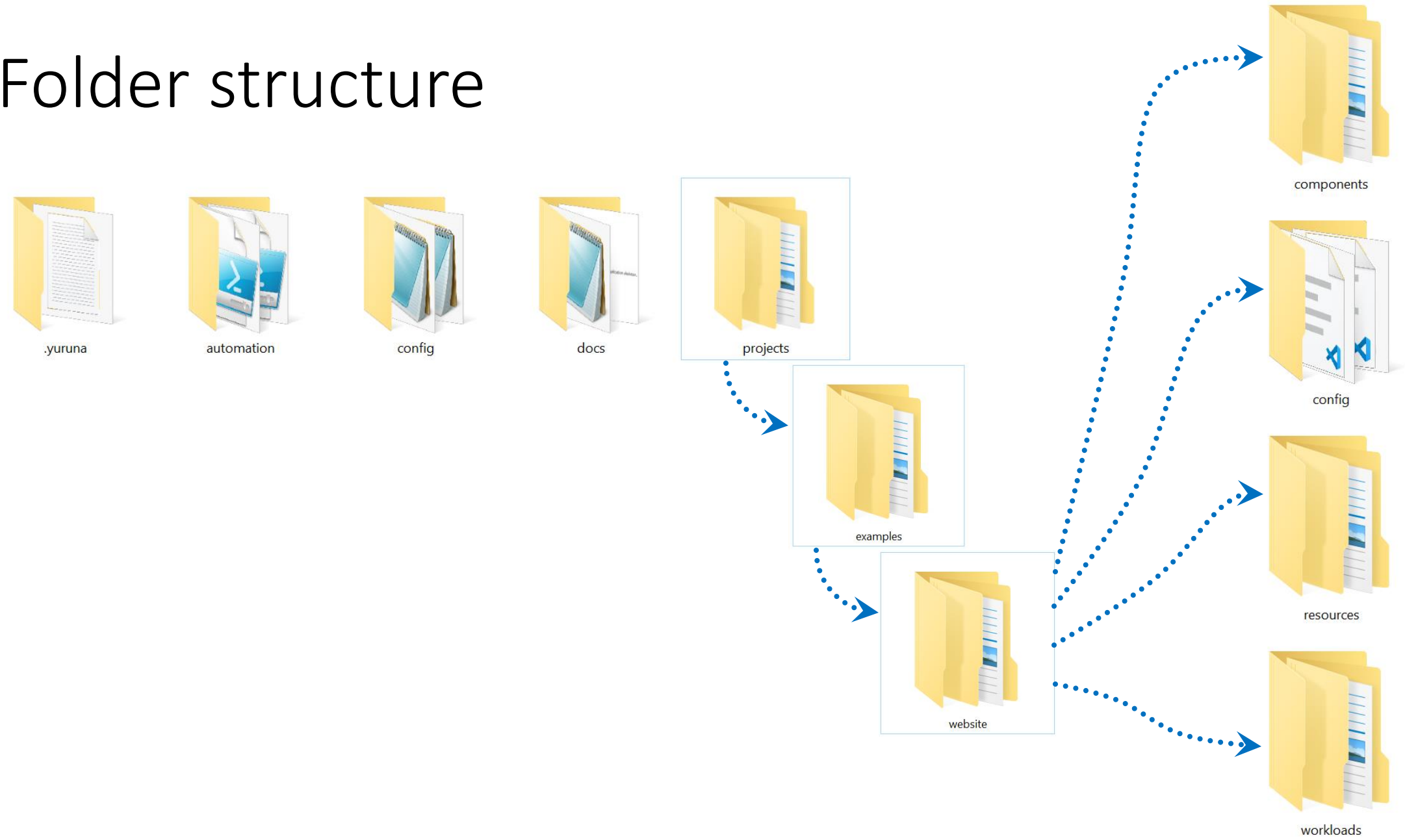
Yuruna goals

- 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

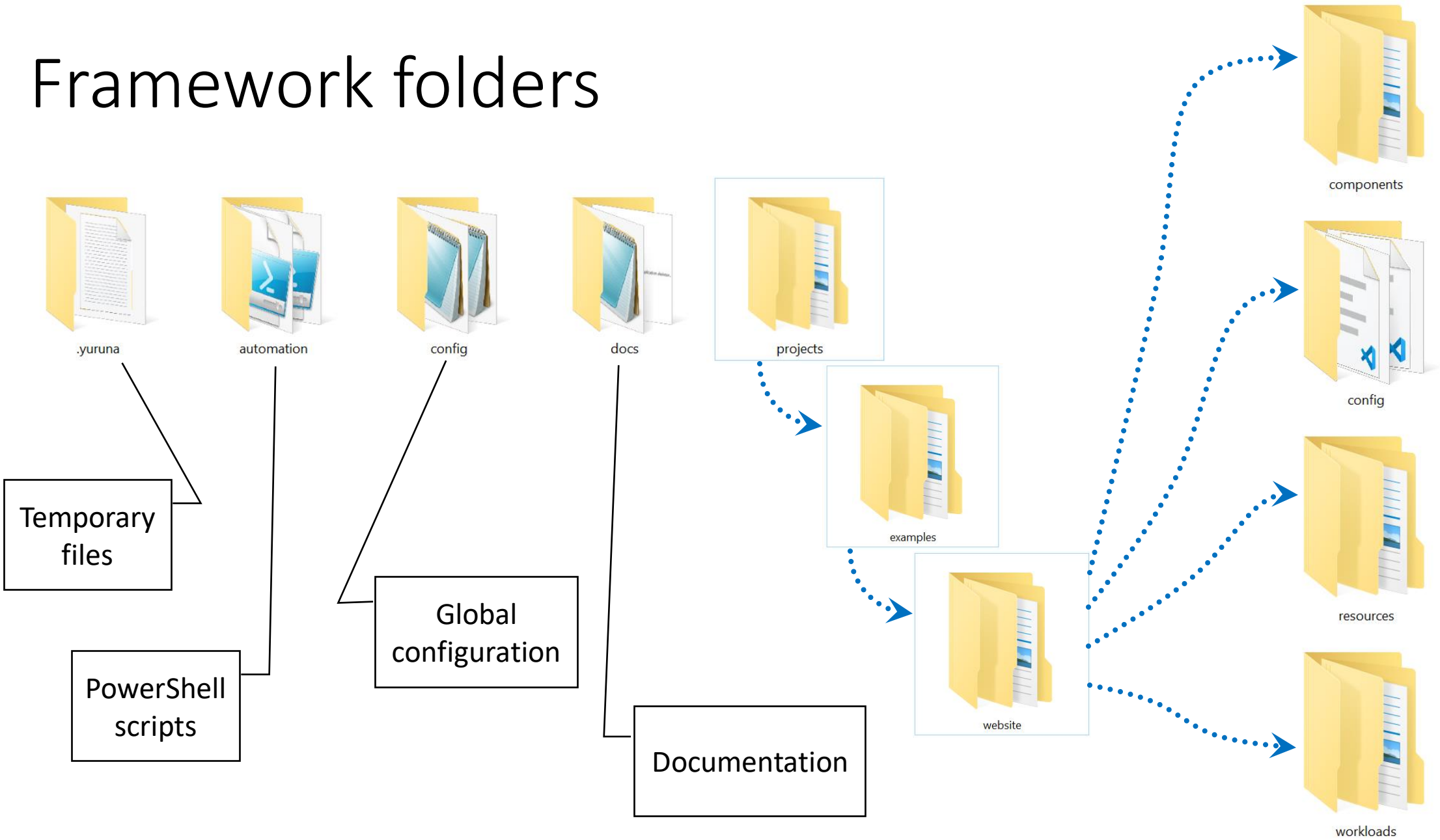
Personas

- Developer using “applications”
  - Just deploy application to their own resources created in their accounts
- Developer configuring “applications”
  - Will pick resource templates and components, and configure how those are deployed as workloads
- Developers creating artifacts
  - Create a configurable Terraform template
  - Create a component
  - Create a Helm chart

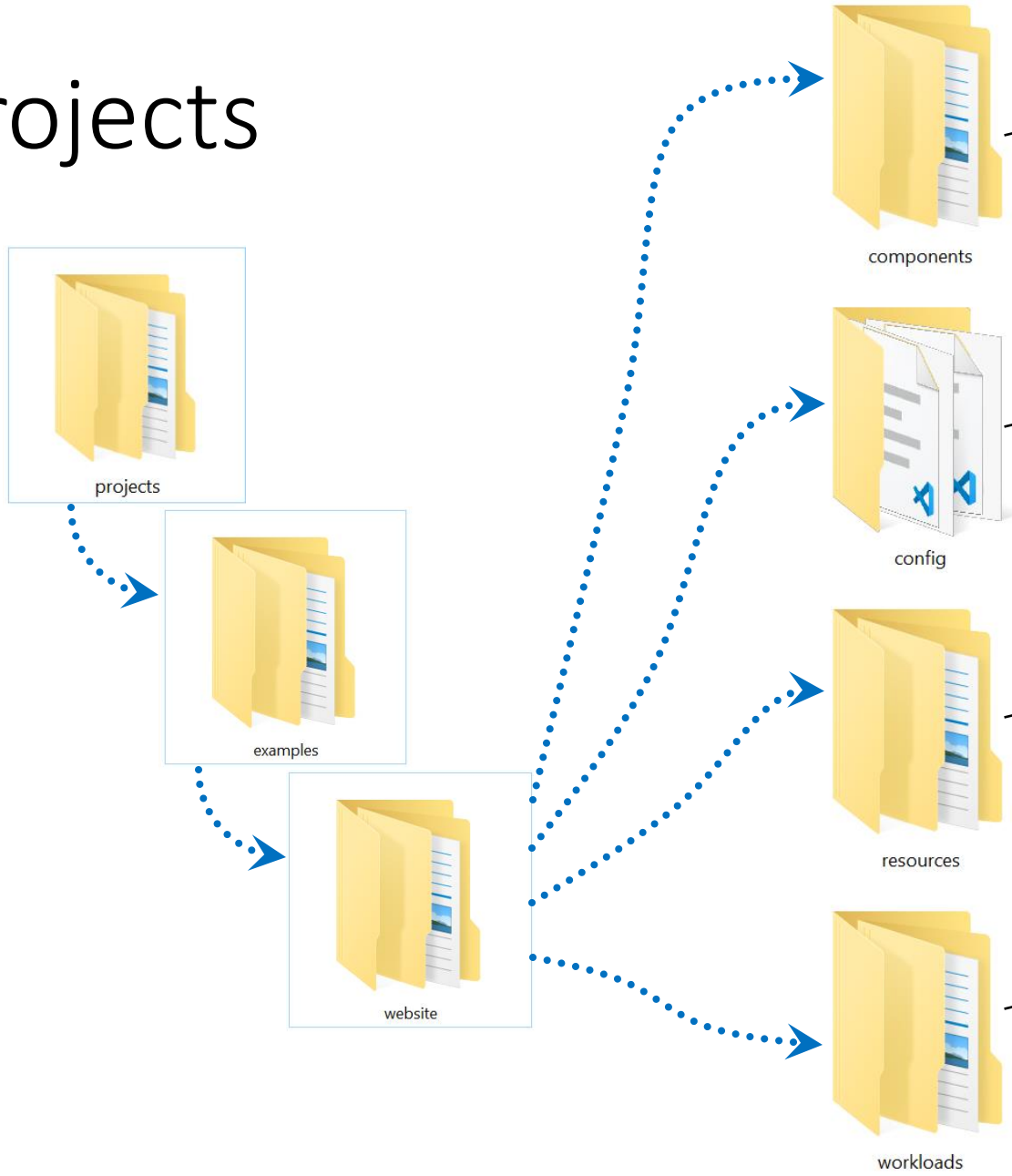
Folder structure



Framework folders



Projects



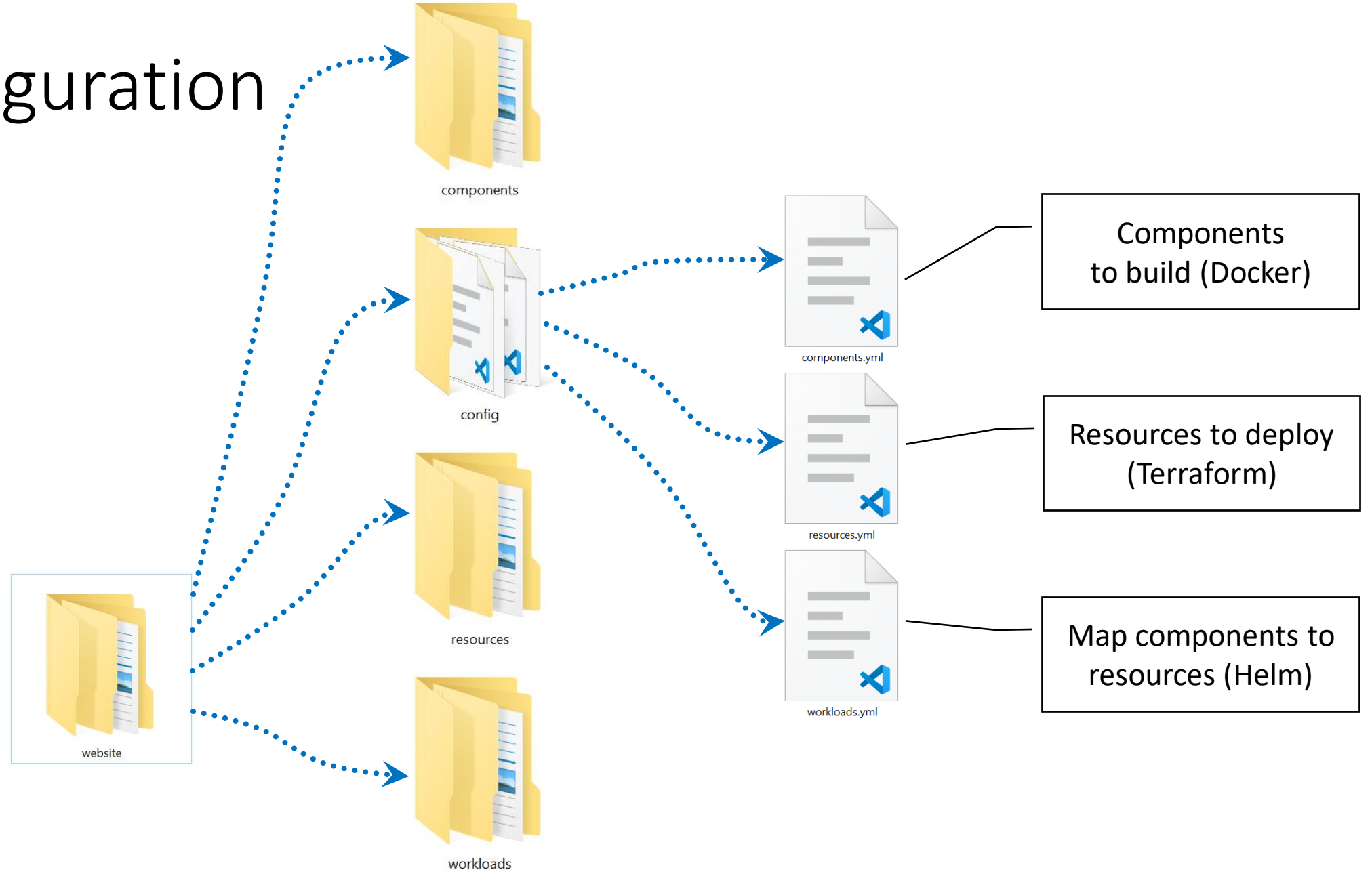
Components:
code and Docker files

Configuration:
project-specific

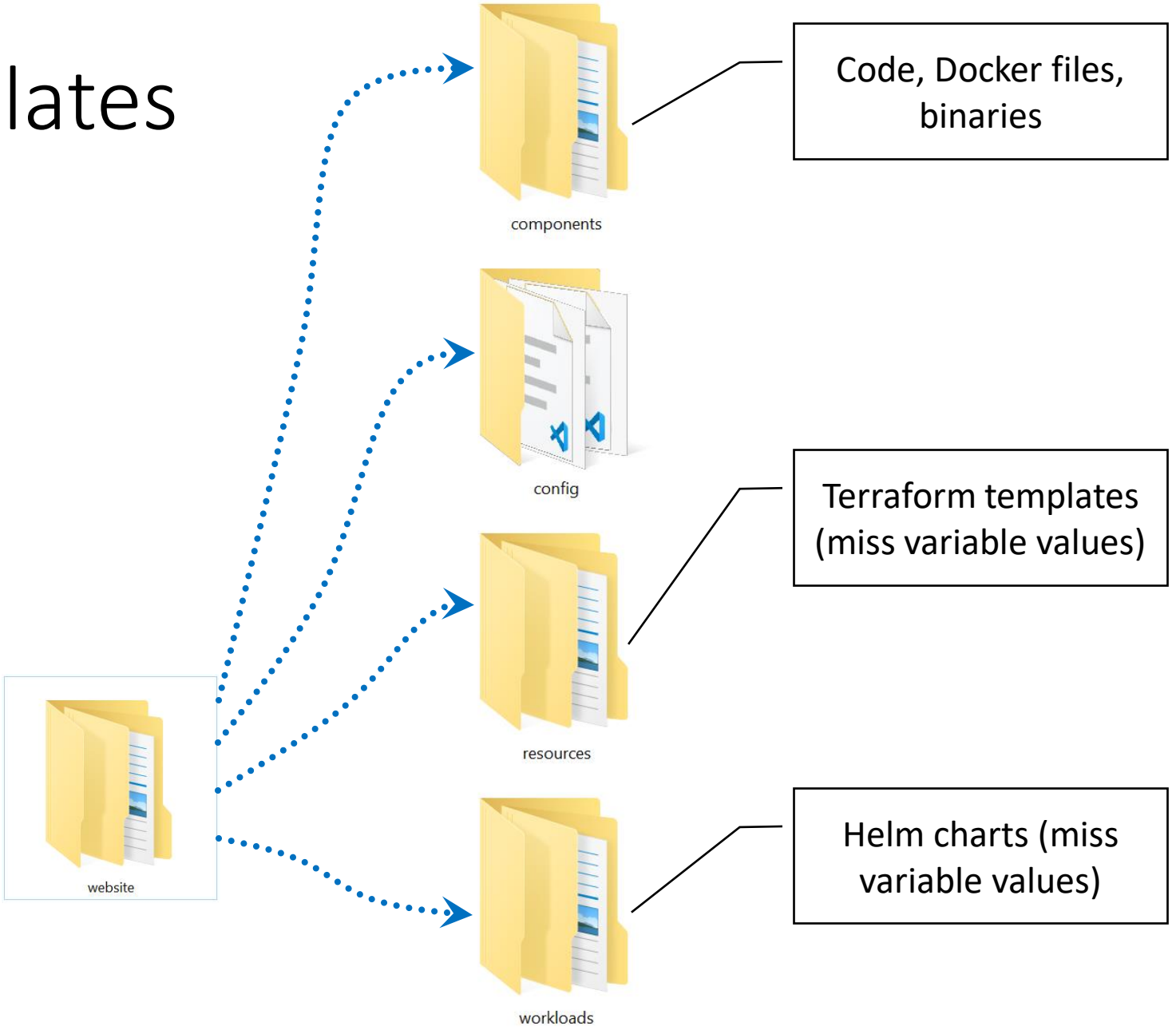
Resource
templates: Terraform

Workloads templates:
Helm

Configuration

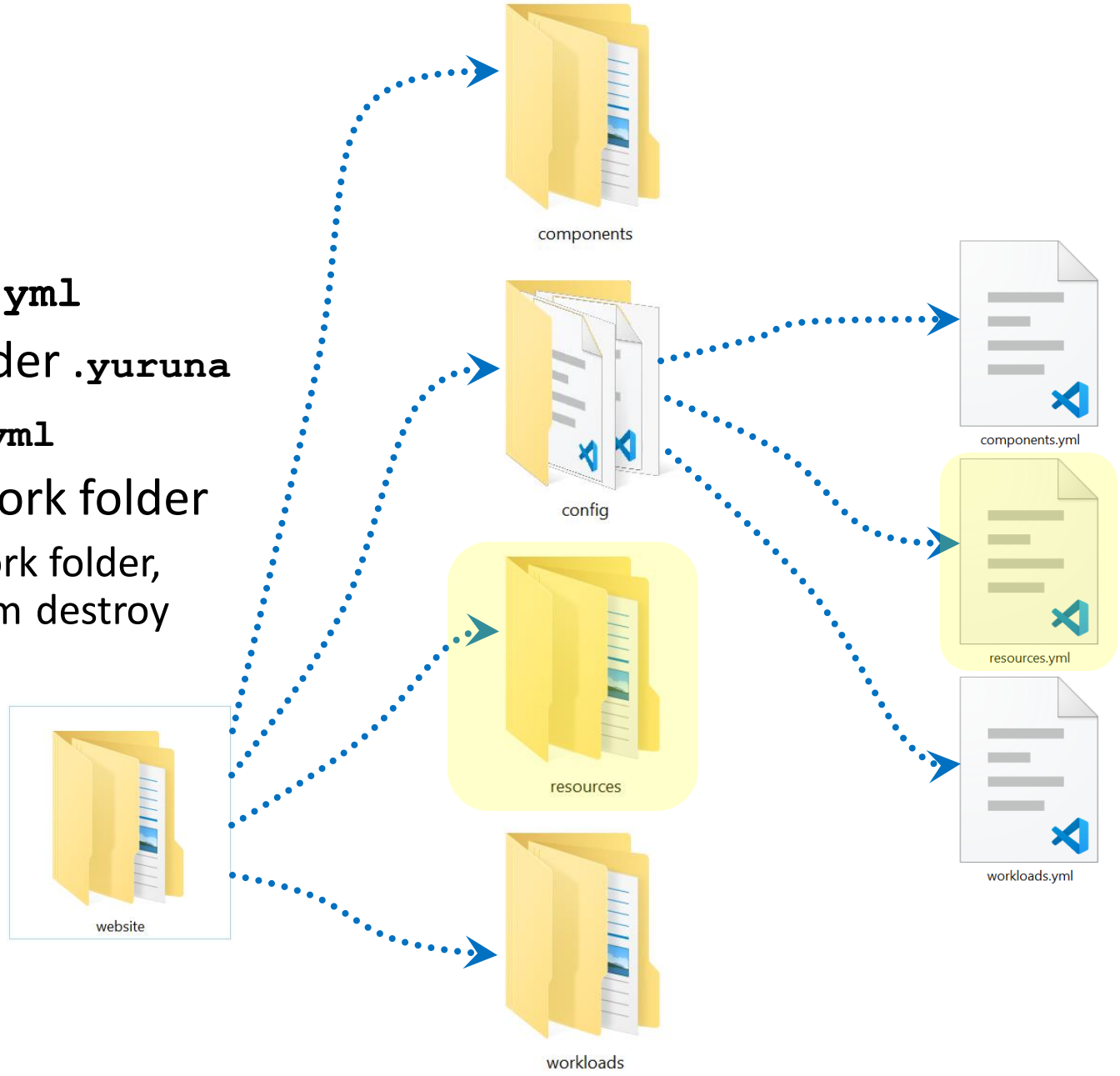


Templates


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

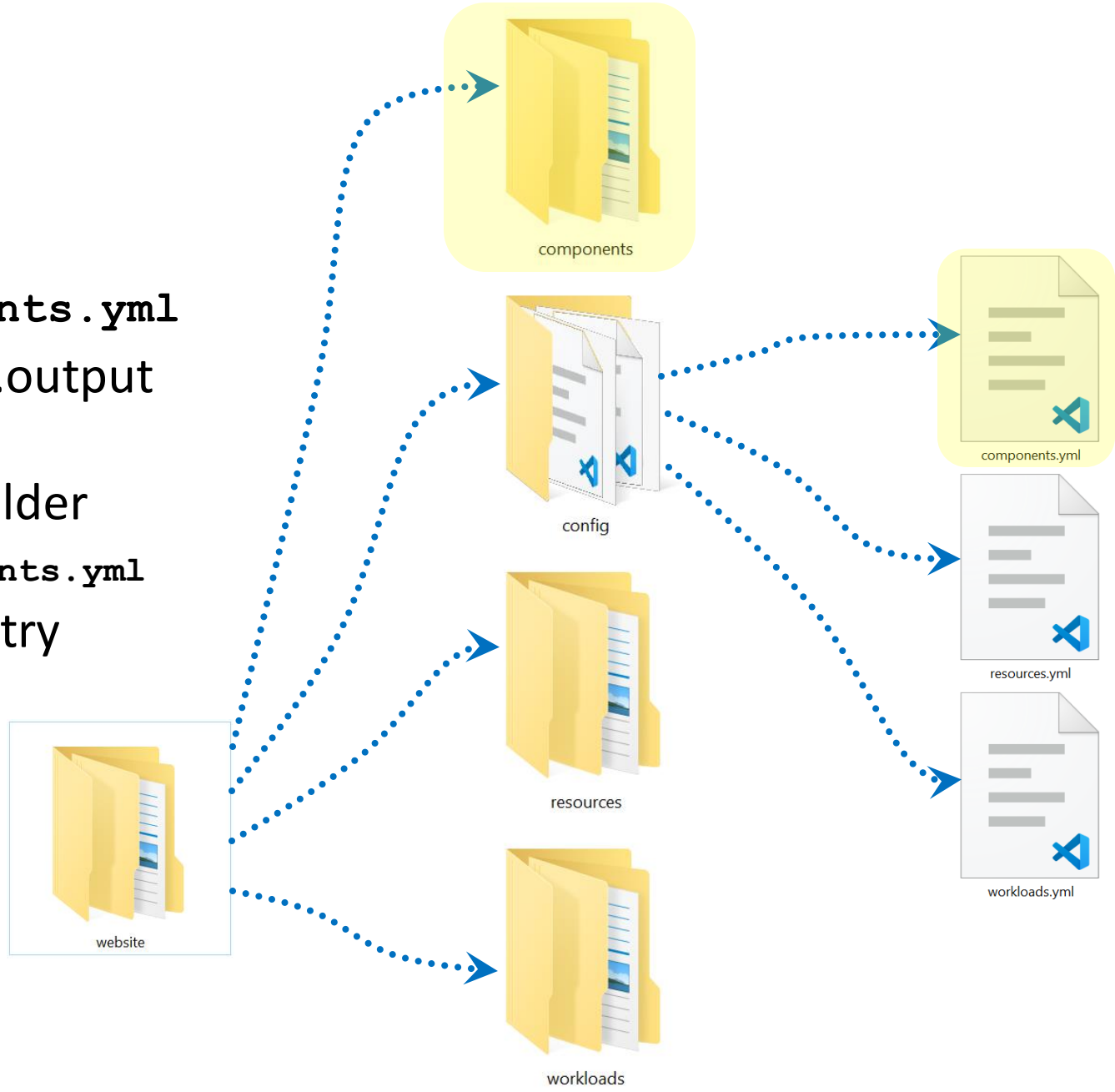
`yuruna resources`

- 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



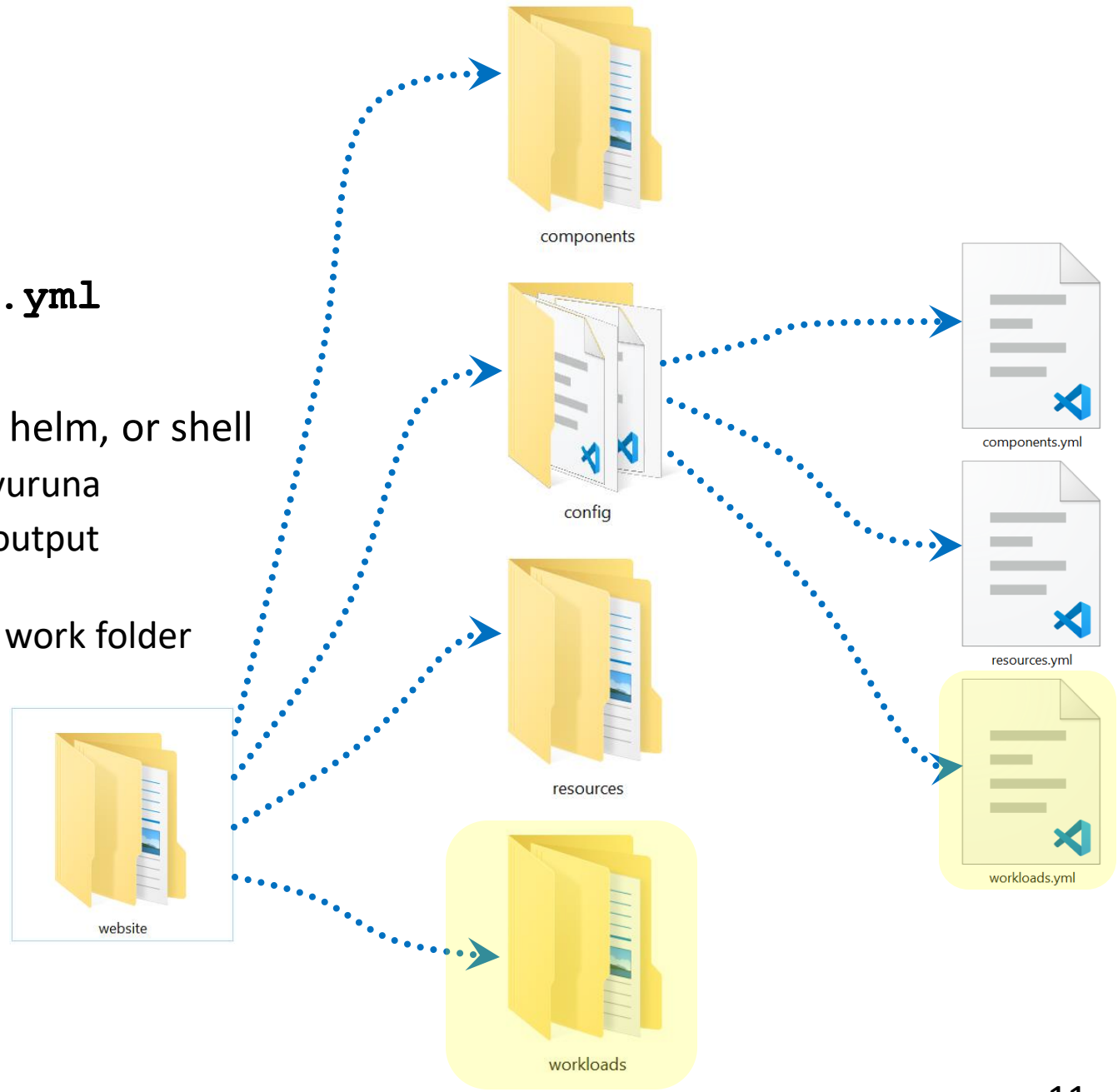
`yuruna components`

- For each component in `components.yml`
  - apply global variables, resources.output variables, workload variables
  - execute build command in the folder
    - command is parameter in `components.yml`
  - tag and push component to registry



`yuruna workloads`

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

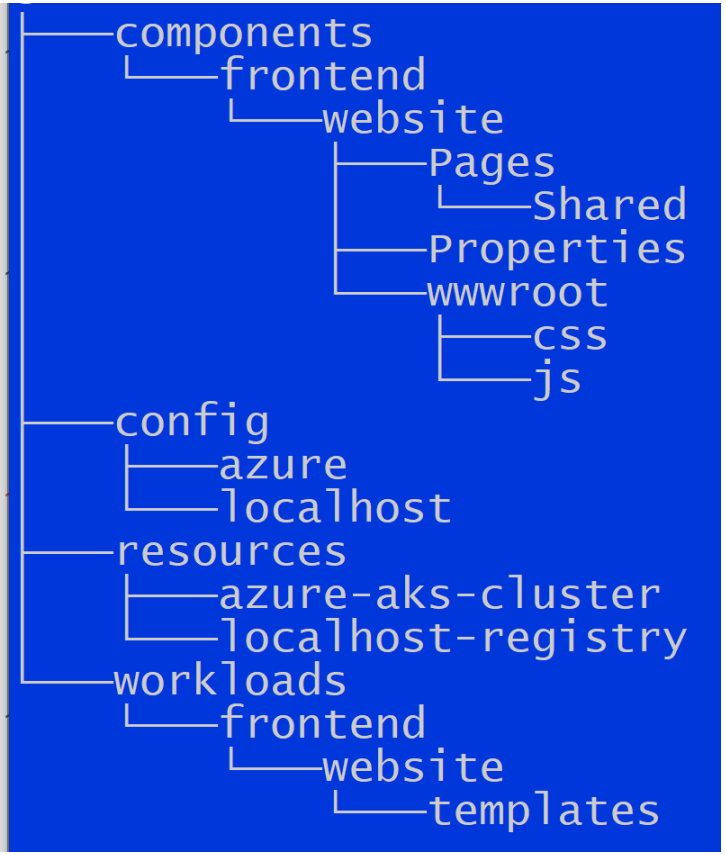


yuruna command details

- 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 [website-folder] localhost`**
  - **`yuruna components [website-folder] localhost`**
  - **`yuruna workloads [website-folder] localhost`**



Resources.yml

```yaml
# Resources information - localhost
---
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:
# Terraform template will run local registry
- name:             "localhost-registry"
  template:         "localhost-registry"
  variables:
```

Components.yml

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

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

Workloads.yml (part 1)

```yaml
# Workloads information - localhost
---
globalVariables:
  namespace:          "yuruna"
  containerPrefix:    "yuruna"
  registryLocation:   "localhost:5000"
  registryName:       "localhost"
  frontendIpName:     "localhost"
  frontendIpAddress:  "127.0.0.1"
  site:               "localhost"
  dockerUsername:     "dummy"
  dockerPassword:     "dummy"
  certManagerIssuerEmail: "certificates@yuruna.com"
  websiteTlsSecret:   "website-tls-secret"
```

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:frontendIpAddress}"
      --set controller.service.annotations."service\.beta\.kubernetes\.io/azure-dns-label-name"="${env:frontendIpName}"
      --set controller.service.annotations."kubernetes\.io/ingress\.global-static-ip-name"="${env:frontendIpName}"
  - kubectl: "apply --validate=false -f https://github.com/jetstack/cert-manager/releases/download/v1.1.0/cert-manager.yaml"
  - shell: "mkcert -install"
  - shell: "mkcert -key-file \"${env:workFolder}/website-tls.key\" -cert-file \"${env:workFolder}/website-
tls.crt\" yuruna.com \"*.yuruna.com\" yuruna.test localhost 127.0.0.1 ::1"
  - kubectl: "delete secret ${env:websiteTlsSecret}"
  - kubectl: "create secret tls ${env:websiteTlsSecret} --key \"${env:workFolder}/website-tls.key\" --cert \"${env:workFolder}/website-tls.crt\""
  - shell: "Start-Sleep -s 59"
  - 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/