R Shiny in the Cloud – Automating the Process with Extensive Collaboration

PHUSE SDE 2025 - FRANKFURT

Abu Zafar Mohammad Sami, *Principal Statistical Programmer*

www.mainanalytics.de

Deployed and Developed a Shiny App: Next Challenges

ui <- fluidPage(

titlePanel(*<your title>*)

sidebarLayout

sidebarPanel(*<input widgets>*)

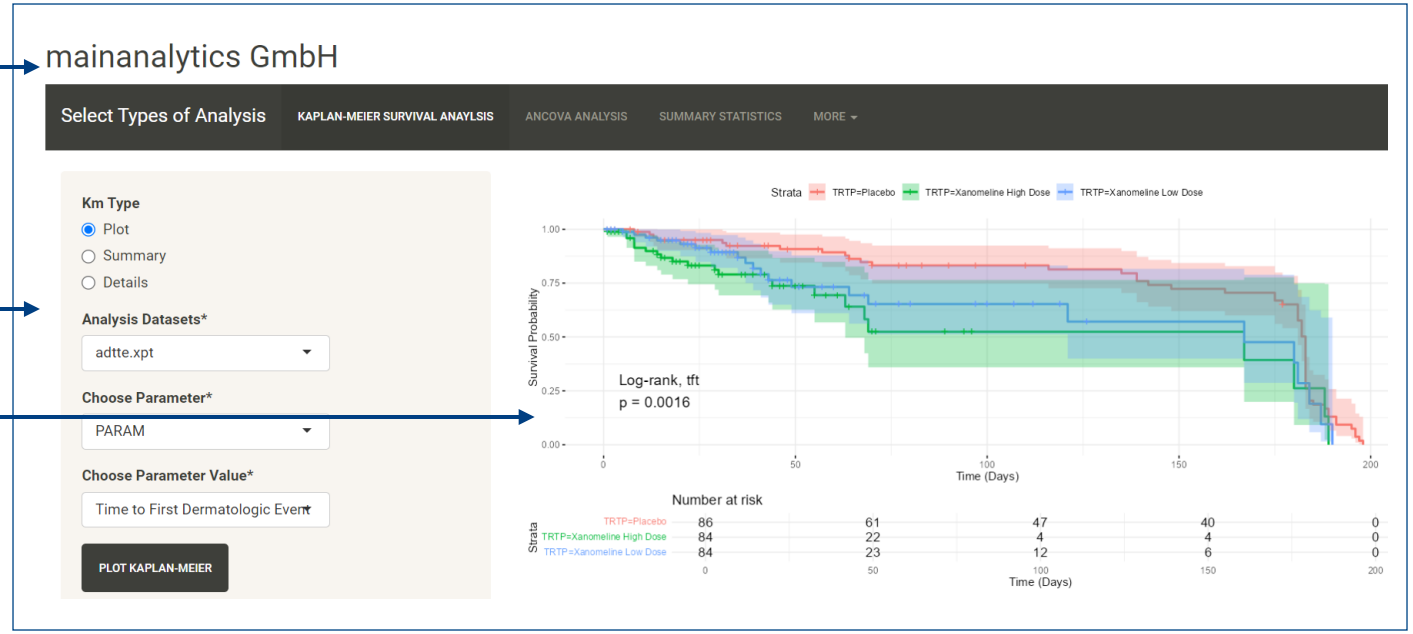
mainpanel (*<output>*)

server <- function(input, output,server) {

<calculations>

}

shinyApp(ui=ui, server=server)



fit <- survfit(<extract parameters from user selections>)
ggsurvplot <- (fit, <draw plot>)

The Challenges After Developing and Deploying a Shiny App

- Validate Packages: Ensure all utilized R packages are appropriate and functioning as intended.
- Statistical Review: Conduct a thorough review of statistical methods and validate results for accuracy.
- Enhance Test Coverage: Increase the scope and depth of tests to ensure robust functionality of the application.
- Streamline CI/CD: Simplify the complexity of continuous integration, development, and deployment processes for better efficiency.

Imposing good software engineering practices

main analytics

Agenda

- Shiny App in Cloud
  - Prerequisites
  - Continuous Integration
  - Continuous Delivery and Deployment
  - Cloud Scripting
- Examples
- Conclusion

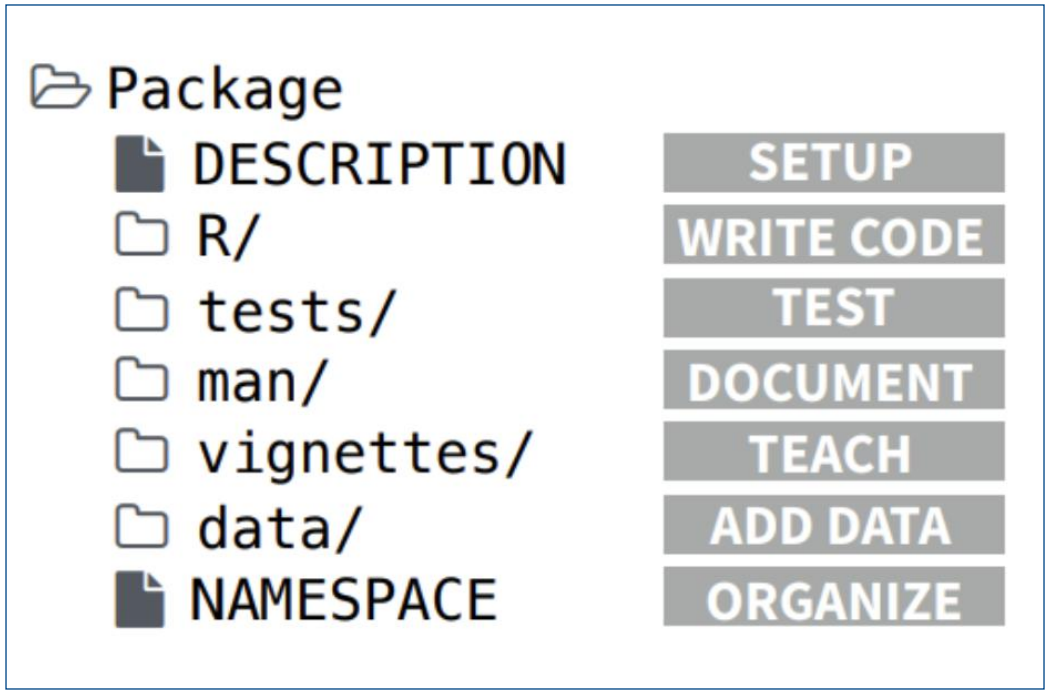
main analytics

Shiny App in Cloud

Prerequisite: Follow Standard R Package Convention

There are multiple packages useful for package development, incl. ***usethis*** (handily automates repetitive tasks).

The 7 common parts of an R Package



Shiny App in Cloud

Prerequisite: Package - renv (Reproducible Environments)

System library

Contains all installed packages

Project library

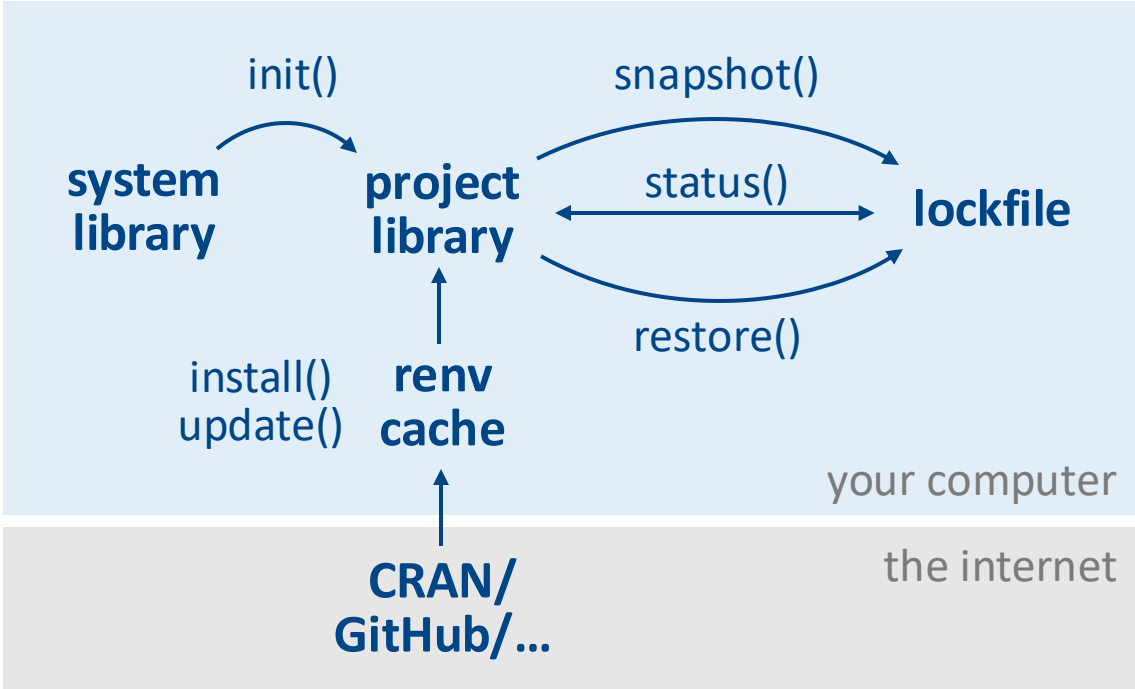
Gives each project its own independent collection of packages

Lockfile

Records metadata about packages to re-install them on a new machine

renv cache

Uses package cache, is shared across all projects



Shiny App in Cloud

Prerequisite: Package - renv (Reproducible Environments)

Advantages

- Resolves version control problem
- Installing and restoring packages is much faster, as renv can find and re-use previously installed packages from the cache

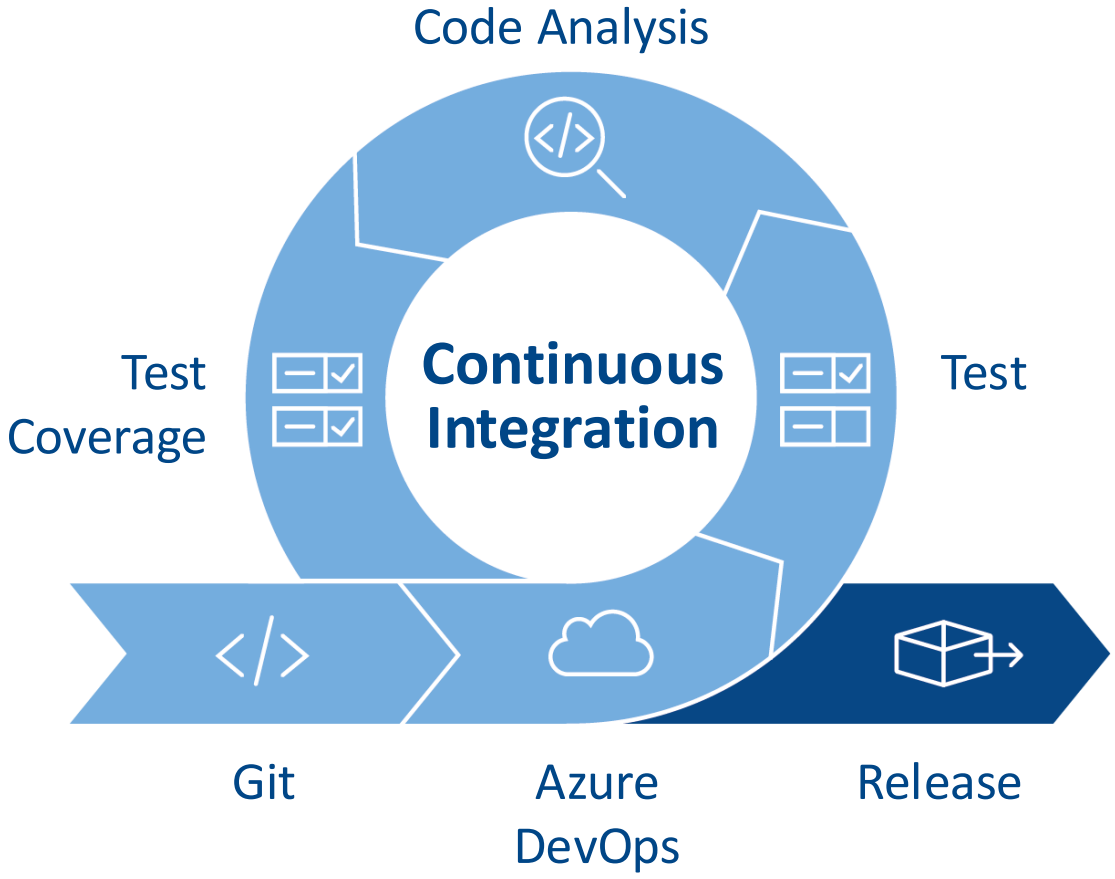

```
{
  "R": {
    "Version": "4.2.2",
    "Repositories": [
      {
        "Name": "CRAN",
        "URL": "https://cran.rstudio.com"
      }
    ]
  },
    "admiral": {
      "Package": "admiral",
      "Version": "1.1.1",
      "Source": "Repository",
      "Repository": "CRAN",
      "Requirements": [
        "R",
        "admiraldev",
        "cli",
        "dplyr",
        "hms",
        "lifecycle",
        "lubridate",
        "magrittr",
        "purrr",
        "rlang",
        "stringr",
        "tidyr",
        "tidyselect"
      ],
      "Hash": "a06f94a38ce50c4745ac7102f161bea4"
    },
```


Example: Lockfile

Shiny App in Cloud

Continuous Integration

- Merged code changes into a central repository
- Automatically builds, tests and validates the leveraged code



CI addresses bugs earlier and quicker, ensures code quality and reduce the time it takes to validate and deployment of new software updates.

Shiny App in Cloud

Continuous Delivery and Deployment (CD)

- After passing all CI automated tests, the application can be deployed at any time through a configured automated release process.
- The product is stored as a container image in a private Azure Container Registry. Azure App Service then automatically pulls the latest container image during deployment, allowing authorized collaborators to interact with the Shiny app, facilitating continuous deployment.

CD enables faster and more frequent releases with less/no manual intervention and ensures consistent deployment across environments and reducing the risk of human error.

Shiny App in Cloud

Cloud Scripting: Package - renv (Reproducible Environments)

Configure the YML-File

```
- script: |
    R -e "install.packages('renv')"
  displayName: 'Install R Packages with renv'

- task: Cache@2
  inputs:
    key: 'renv | "$(Agent.OS)" | renv.lock'
    path: '~/.cache/R/renv'

- script: |
    R -e "renv::restore()"
  displayName: 'Restore R Packages with renv'
```

cache renv in cache directory

Shiny App in Cloud

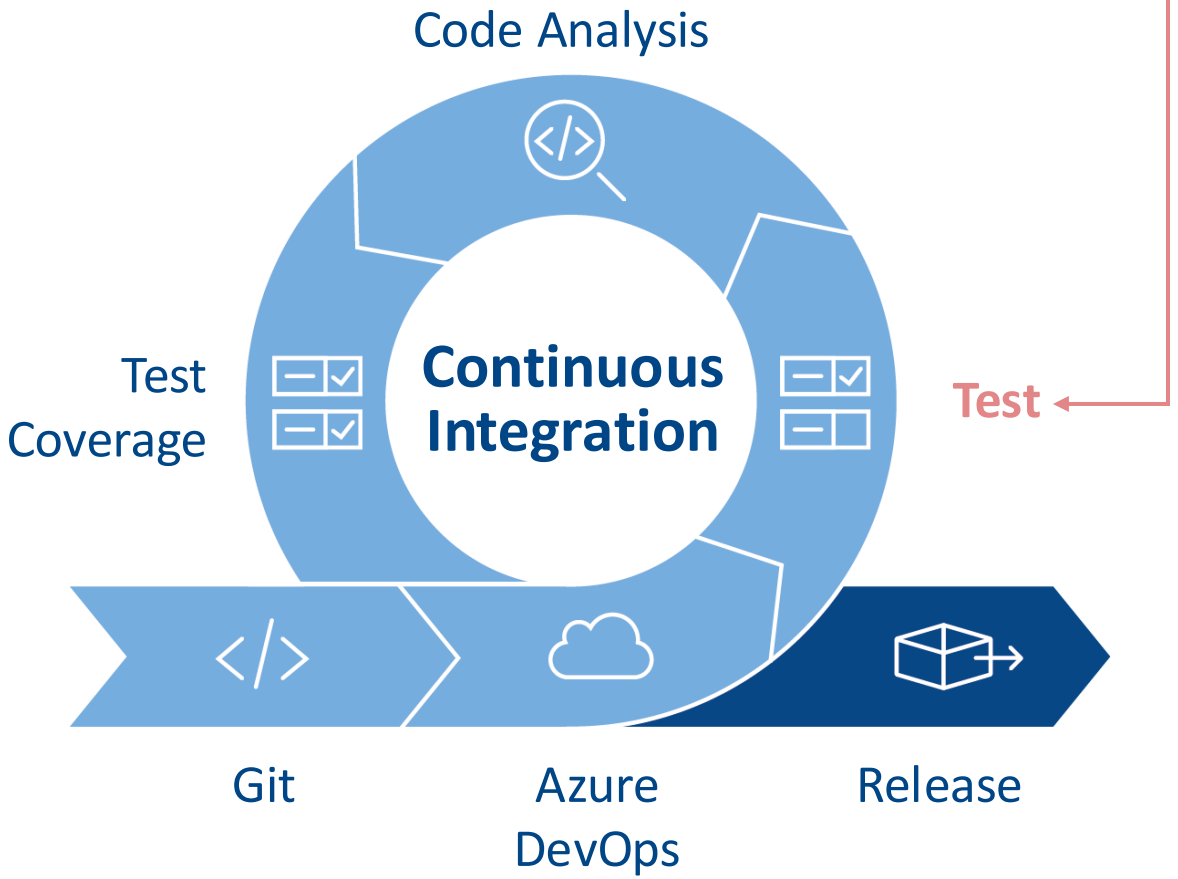
Cloud Scripting

R CMD Checks

CI pipelines run R CMD Checks to verify the quality and structure of an R package

Configure the YML-File:

```
- script: |
  R -e "install.packages('rcmdcheck')"
  R -e "rcmdcheck::rcmdcheck(error_on = 'warning')"
displayName: 'Install and Run R CMD check'
```



Details: https://r-pkgs.org/r-cmd-check.html

Shiny App in Cloud

Cloud Scripting

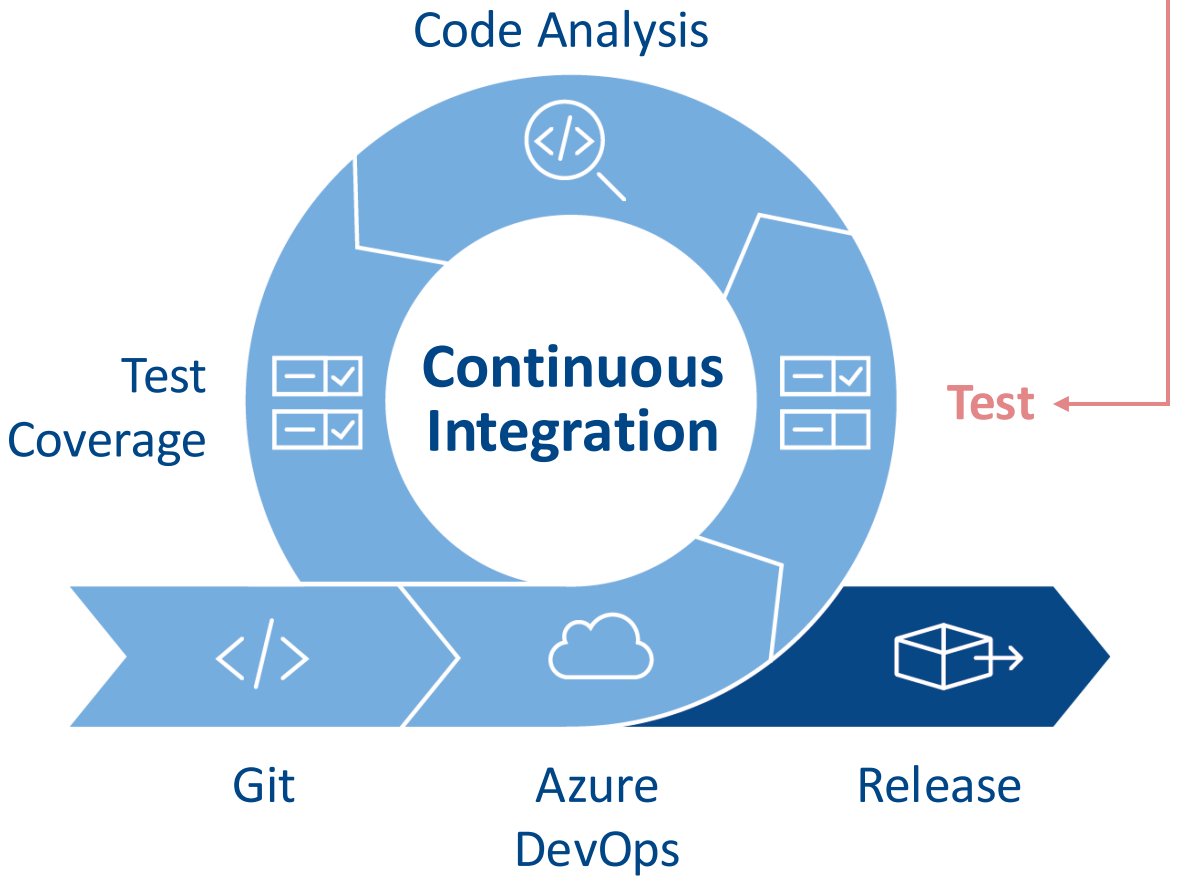
R CMD Checks

Package Structure

- Checking for executable files
- Checking for hidden files and directories.
- Checking for sufficient/correct file permission
- Checking installed package size.
- Checking top-level files.

R code

- Checking R files for non-ASCII characters
- Checking R files for syntax errors
- Checking replacement functions



Shiny App in Cloud

Cloud Scripting

R CMD Checks

Documentation:

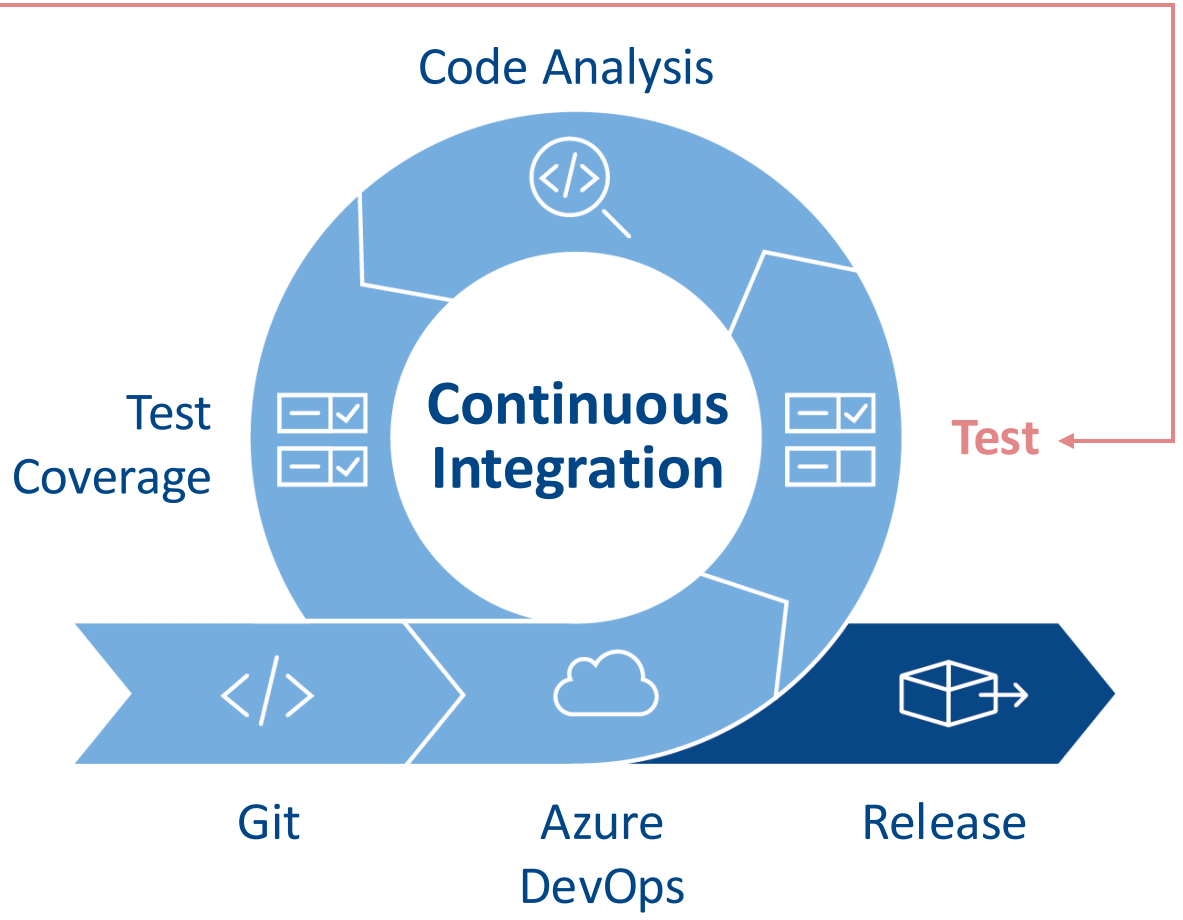
- Checking .Rd files

Tests:

- Checking each file in *tests/* is run

Vignettes:

- Checking running R code from vignettes
- Checking for unstated dependencies in vignettes



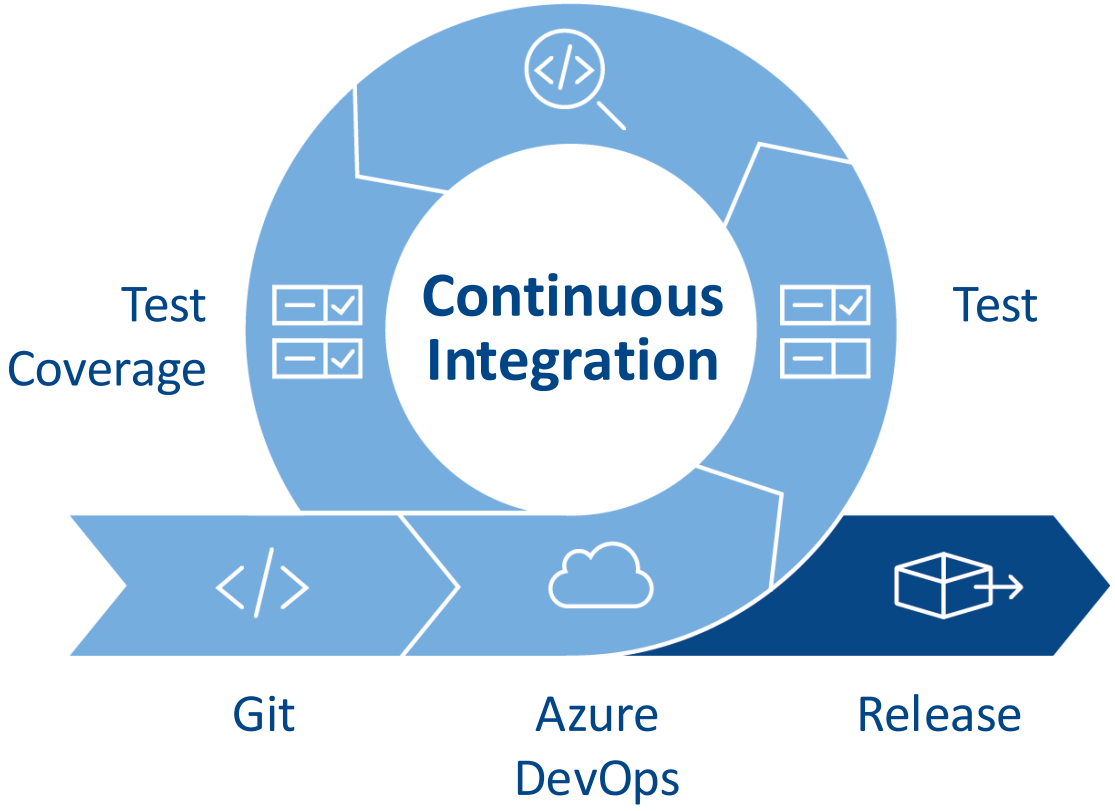
Shiny App in Cloud

Cloud Scripting

oxsecurity/megalinter (lintr package) → **Code Analysis**

On each push request, automatically analyze all code through:

- Syntax Error Detection
- Semantic Issue Identification
- Style Adherence



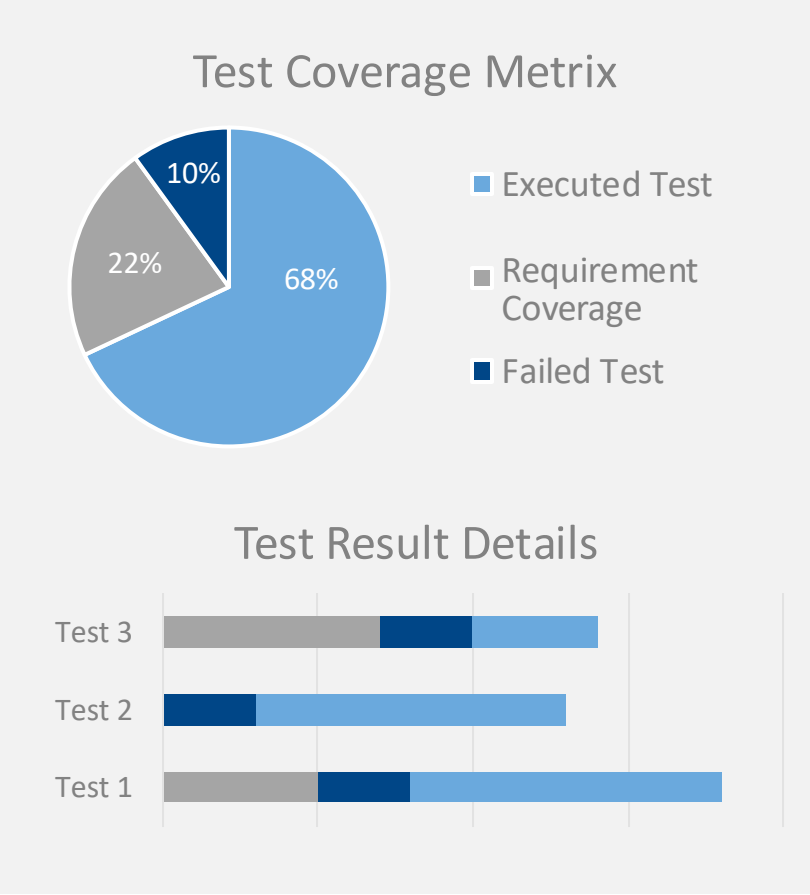
Configure the YML-File:

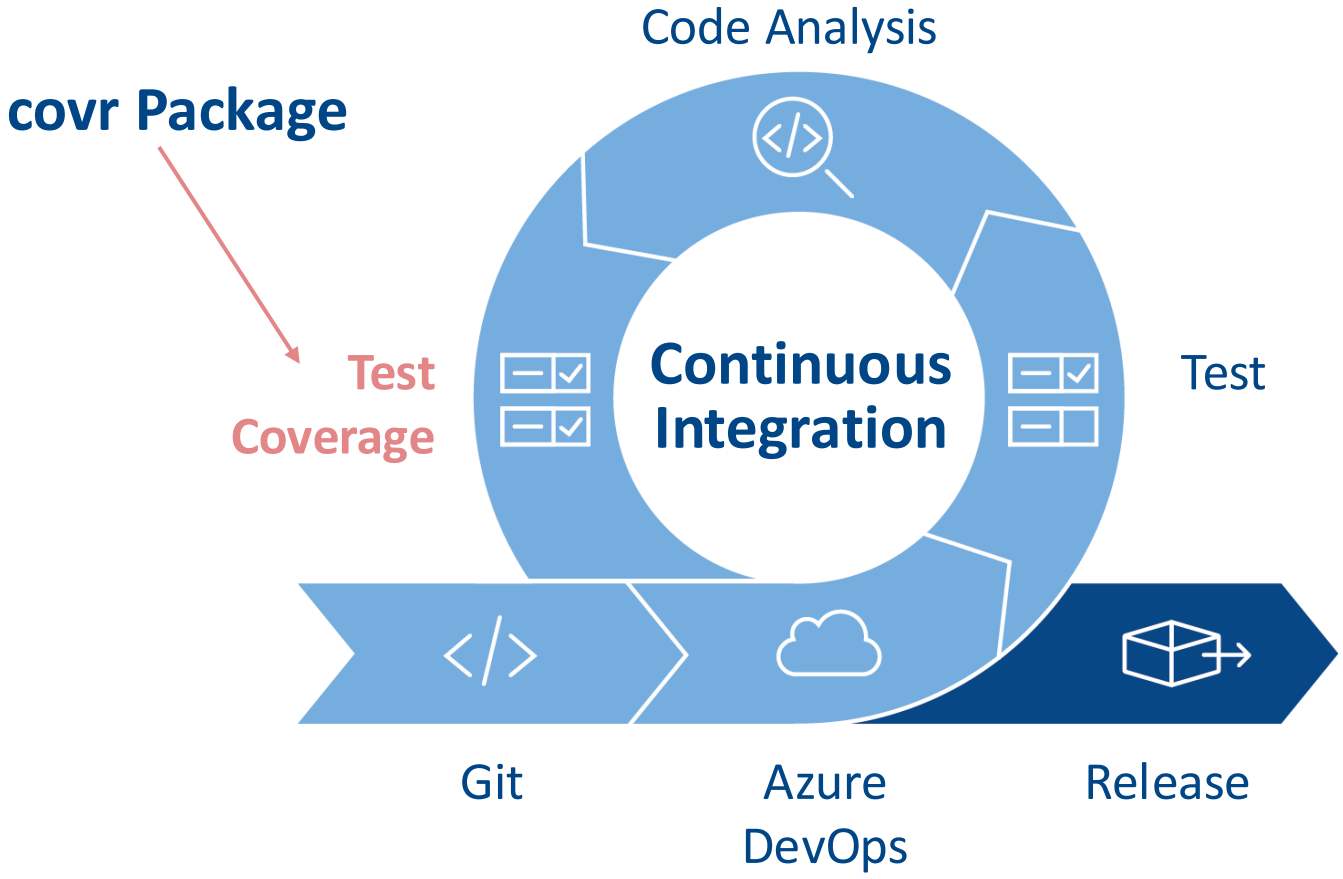
```
- script: |
  R -e "install.packages('lintr')"
  R -e "lintr::lint_package()"
displayName: 'Install and Run lintr'
```

main analytics

Shiny App in Cloud

Cloud Scripting





Examples

ANCOVA Module Output

Type III ANCOVA of Diastolic Blood Pressure (mmHg) by Planned Treatment

adjusted for Baseline Diastolic Blood Pressure (mmHg)

	n	Mean	SE	95% CI	p-value
Placebo					
Model Estimates	72	75	0.88	[74, 77]	
Xanomeline Low Dose					
Model Estimates	60	75	0.96	[73, 77]	
Contrast / Placebo		-0.5	1.3	[-3.1, 2.1]	0.698
Xanomeline High Dose					
Model Estimates	56	76	0.99	[74, 78]	
Contrast / Placebo		0.41	1.3	[-2.2, 3]	0.755
Contrast / Xanomeline Low Dose		0.92	1.4	[-1.8, 3.6]	0.507

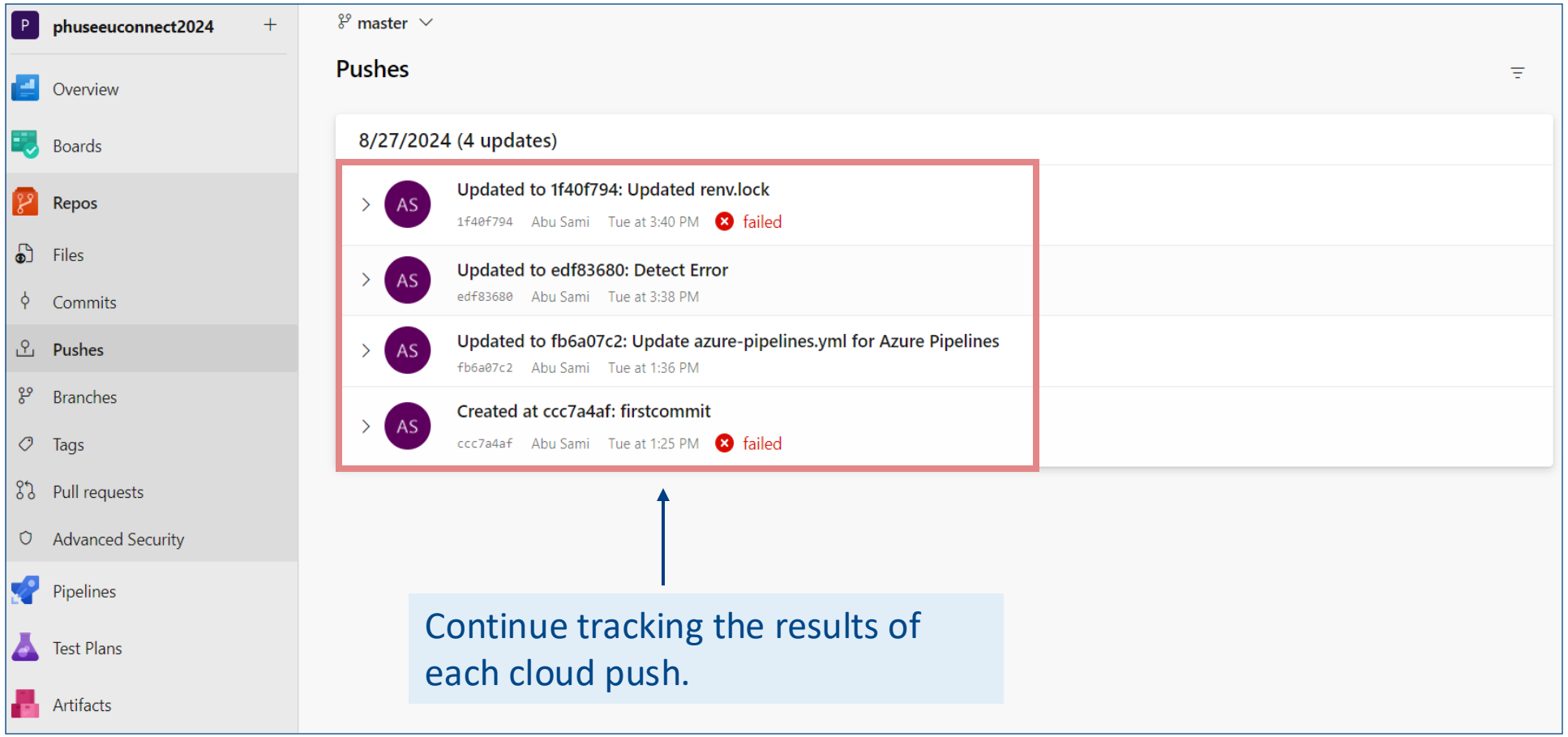
Treatment p-value: 0.800
Covariate p-value: <0.001

A corresponding forest plot should also be added to the Shiny application

main analytics

Examples

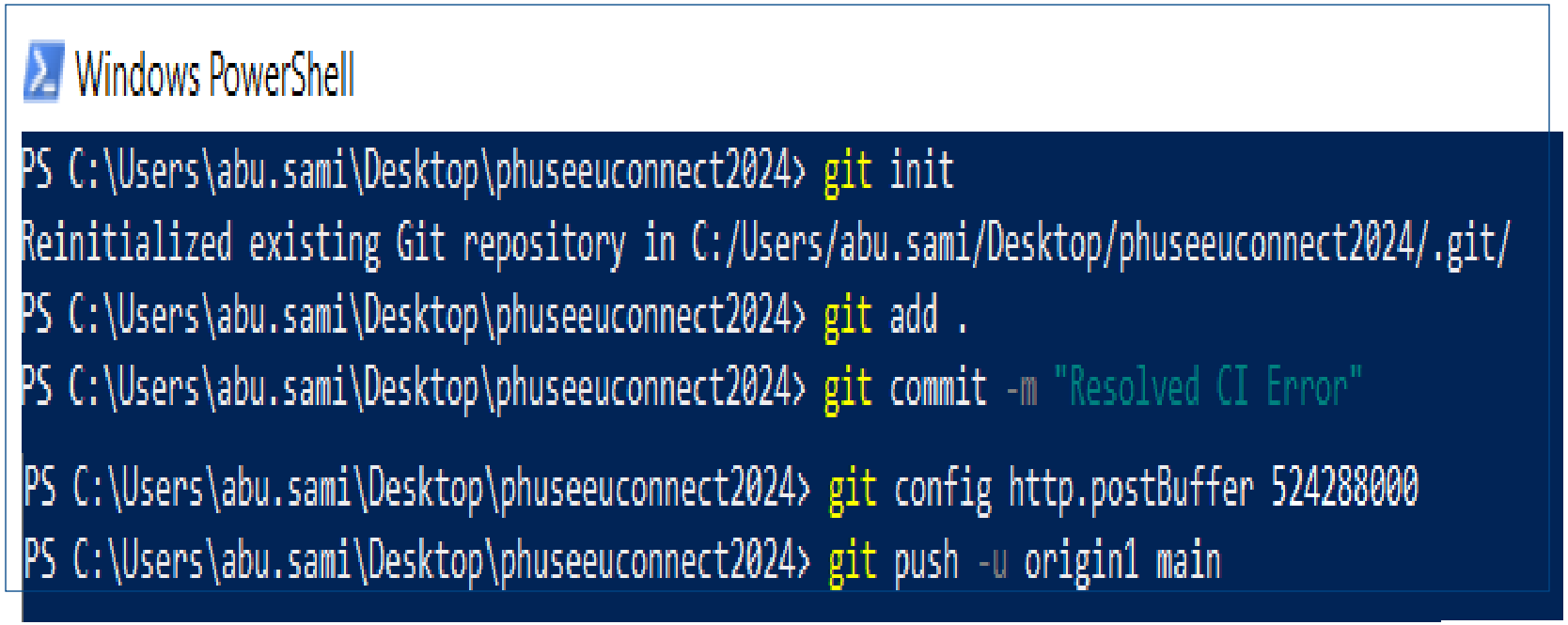
DevOps



Continue tracking the results of each cloud push.

Examples

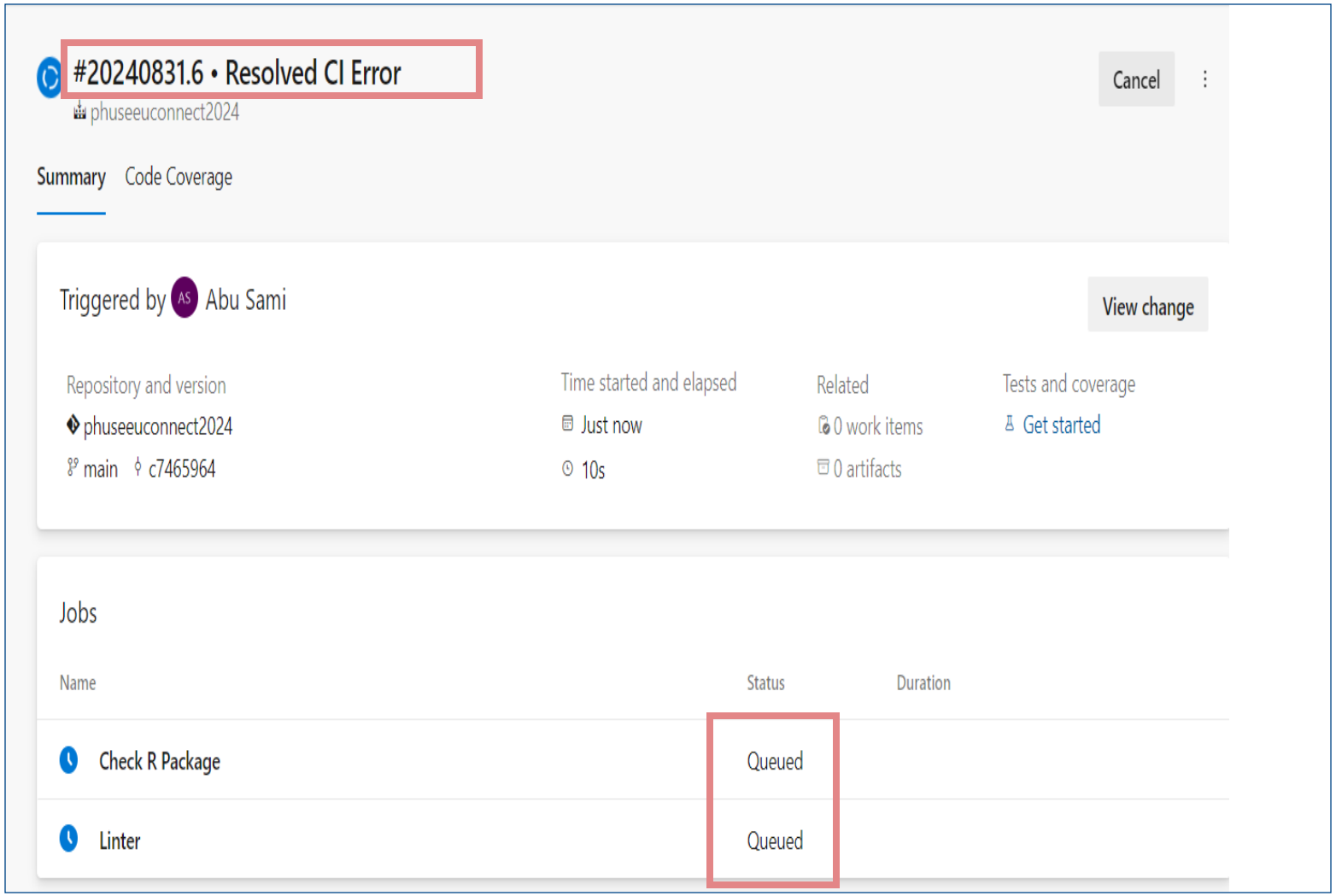
DevOps



Push request of the updated version of the application to the cloud via version control system

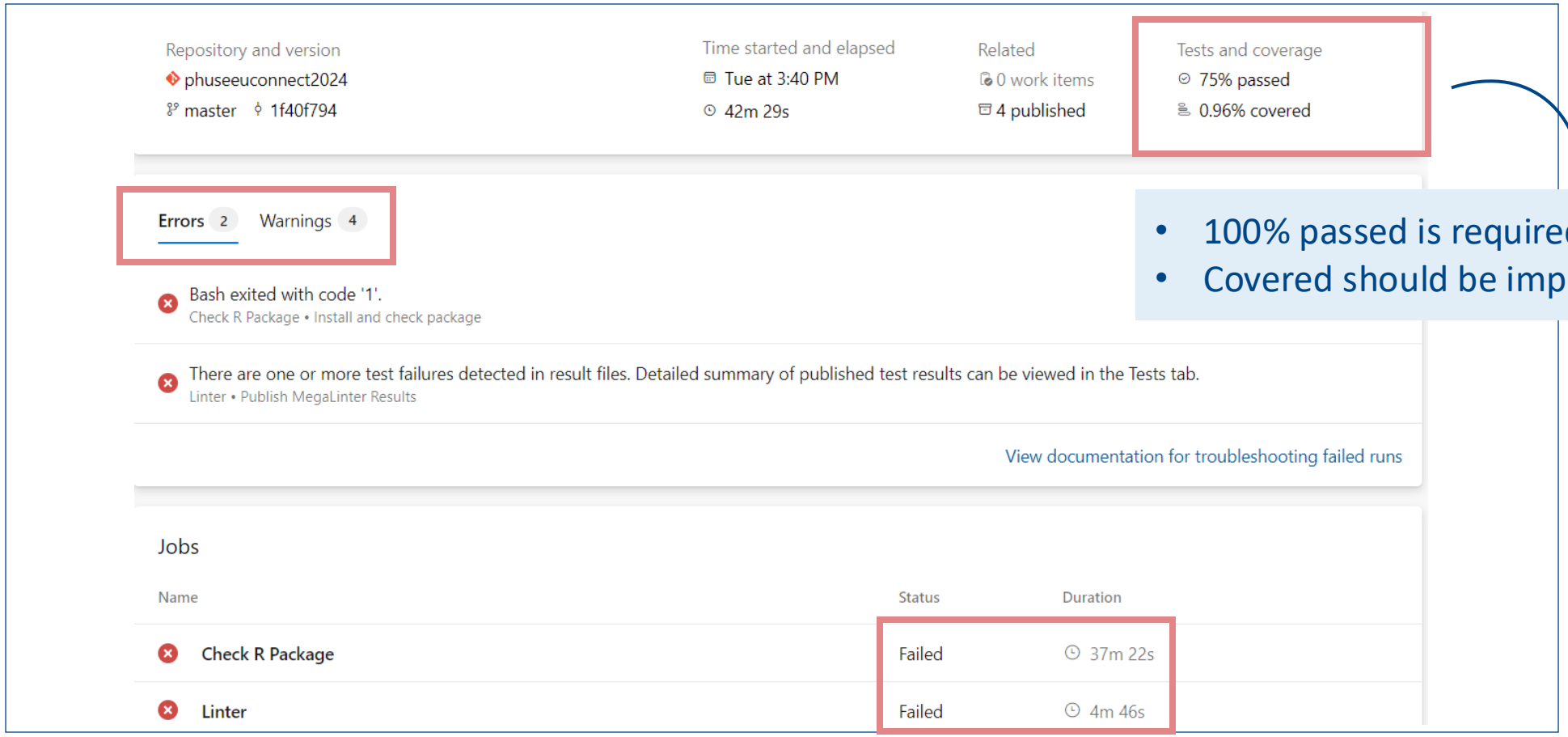
Examples

DevOps



Examples

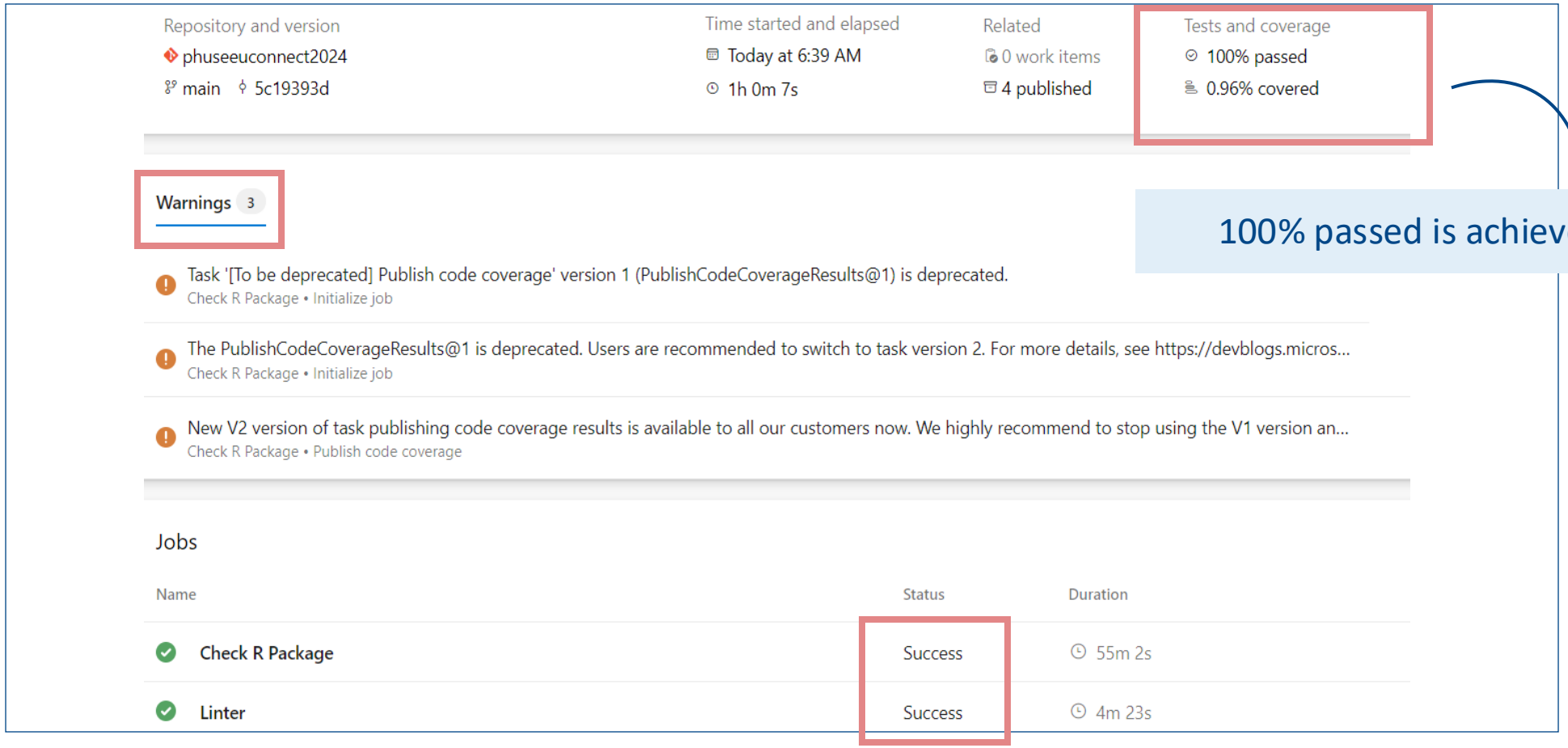
DevOps



- 100% passed is required
- Covered should be improved

Examples

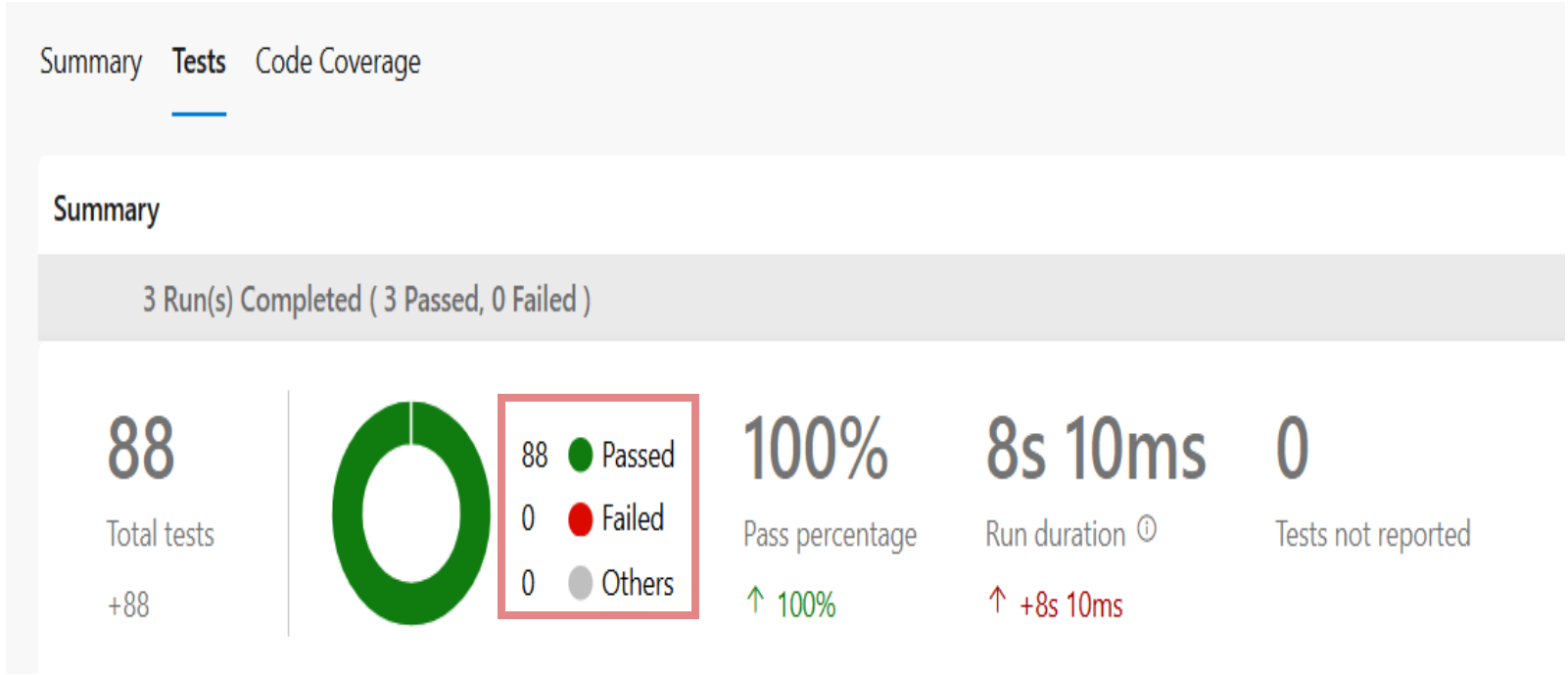
DevOps



100% passed is achieved

Examples

DevOps



Examples

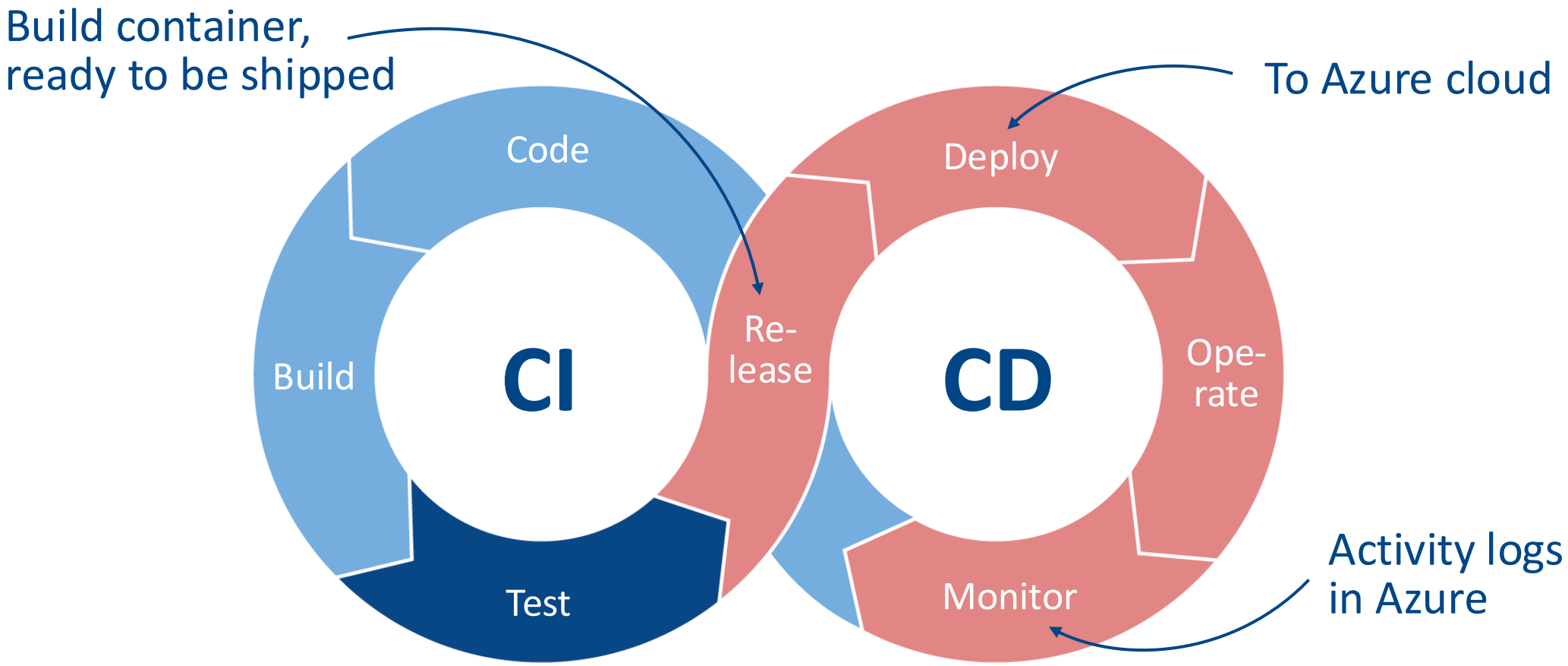
DevOps

Summary Tests Code Coverage

	Line coverage				
Name	**Covered**	**Uncovered**	**Coverable**	**Total**	**Percentage**
shiny.portfolio	**22**	**2272**	**2294**	**3722**	**0.9%**
app-main.R	0	1	1	17	0%
form-engine.R	0	57	57	157	0%
import.R	20	7	27	104	74%
module-dsfilter.R	0	415	415	603	0%
module-form-anova.R	0	160	160	206	0%
module-form-numbycat.R	0	92	92	132	0%
module-form-numbynum.R	0	86	86	126	0%
module-form-surv.R	0	112	112	151	0%
module-stat-anova-ancova.R	0	443	443	556	0%

Example

Ready for Delivery and Deployment (CD)



Conclusion

Strengths

- In response to each update, minimal or no human intervention is required to release a new version
- Foster better collaboration among teams by providing version control, build automation, testing, and deployment, all in one place.

Challenges

- Improvements in unit tests are required. My colleague, Féat Yann, will be discussing this topic, 'Analysis Results Standard for Test-Driven Development,' at the 2025 CDISC+TMF Europe Interchange.

main analytics

Thank you. Any questions?

abu.sami@mainanalytics.de

www.mainanalytics.de