Behind the scene of isc-tar project and story about Continuous Integration using GitHub Actions

Article 

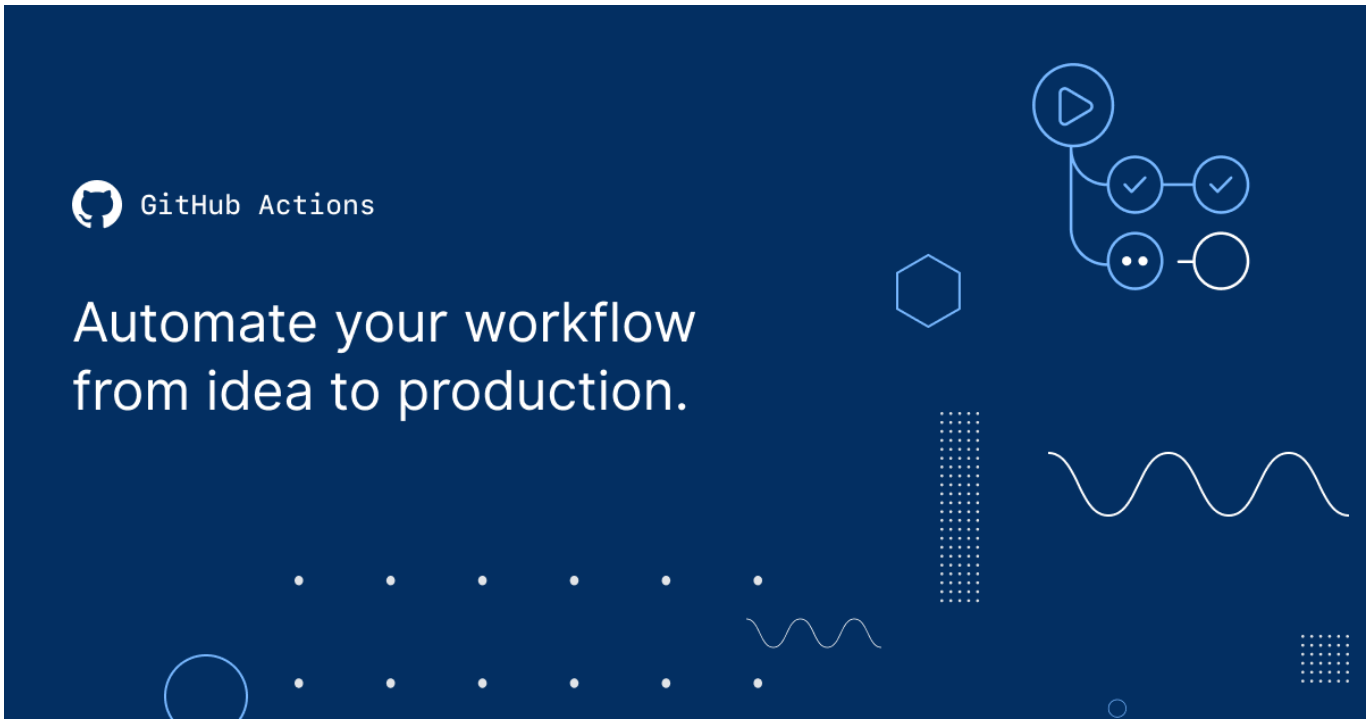
Dmitriy Maslennikov · Mar 20, 2019 3m read

Behind the scene of isc-tar project and story about Continuous Integration using GitHub Actions

This is a continuation of my story about the development of my project isc-tar started in the first part.

Just having tests is not enough, it does not mean that you will run tests after all changes. Running tests should be automated, and when you cover all your functionality with tests, everything should work well after any change in any place. And Continuous Integration (CI) helps to keep the code and deployment procedure with as fewer bugs as possible and automates the routine procedures, like publishing releases.

I use GitHub to store the source code. And some time ago GitHub started to work on its own CI/CD platform and named it GitHub Actions. It is not widely available, yet. You have to be signed as a beta tester for this feature, as I did. GitHub Actions uses quite a different way how to deal with a build workflow. What is important that Github Actions allows to use Docker, and it's quite easy to customize available actions. And interesting that GitHub Actions is really much bigger than any classic CI like we have in Travis, Circle or Gitlab CI and so on. You can find more in the official documentation.



So, let's do it. First of all, I have to run tests by any push changes to GitHub. GitHub actions should be stored in .github folder in file with workflow extension. So, my first version of .github/main.workflow will be like this

```
workflow "Push Workflow" {
```

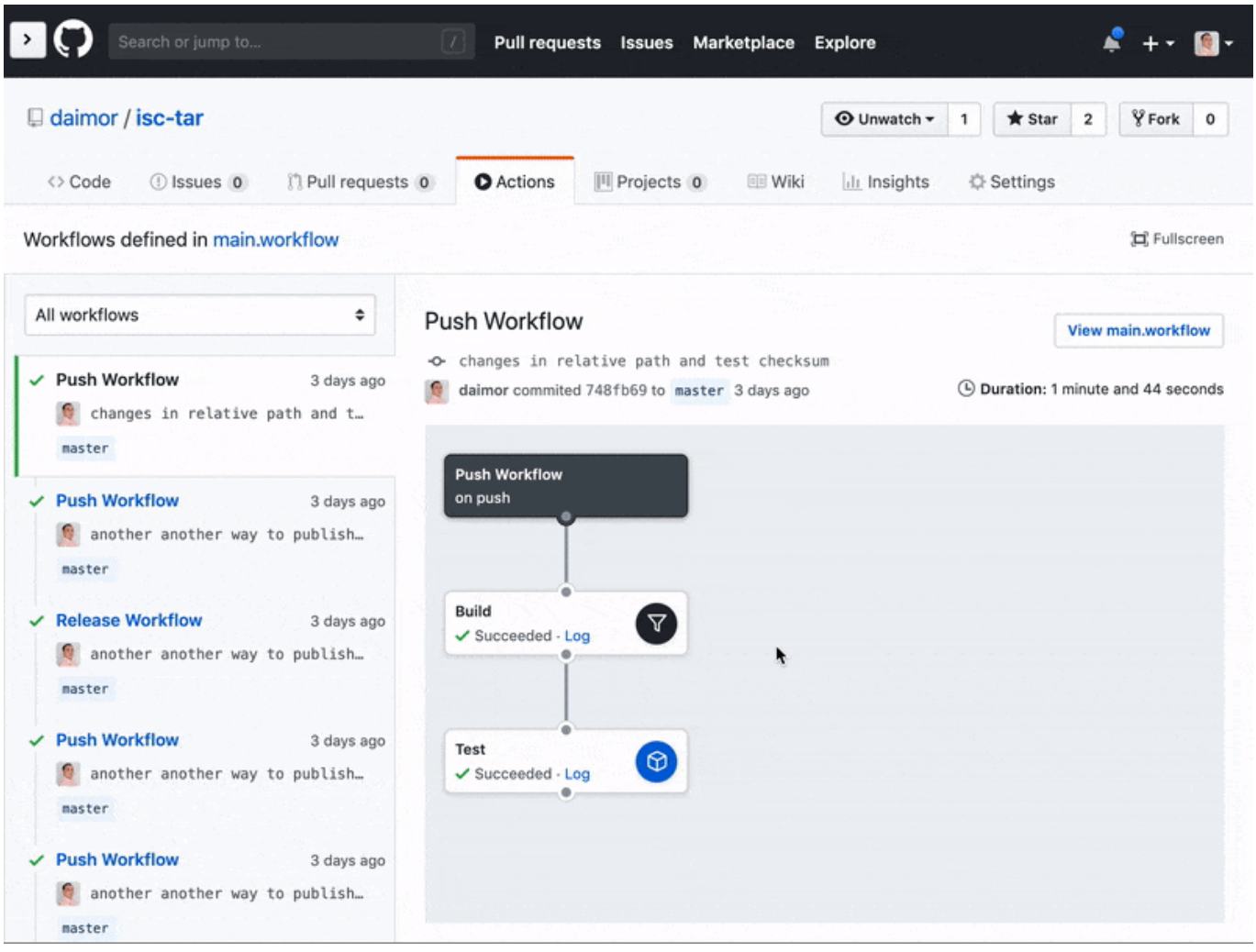
```
  on = "push"
  resolves = ["Test"]
}

action "Build" {
  uses = "actions/action-builder/docker@master"
  runs = "make"
  args = "build"
  env = {
    IMAGE = "daimor/isc-tar"
  }
}

action "Test" {
  needs = ["Build"]
  uses = "docker://daimor/isc-tar"
  runs = "/tests_entrypoint.sh"
}
```

According to the script, I have the workflow which will be activated by push event, with the final action Test, which in meanwhile depends on action Build. Action Build as you may notice, just runs command make with argument build, and predefined IMAGE variable. This action will build docker image tagged as daimor/isc-tar. And after that, we can use this image as an Action with just different entrypoint. GitHub Actions may use any docker image to run action, when it runs it, it mounts your repository to inside the container and defines this mount point as a workdir.

And how this workflow will look in GitHub UI



??????Next, I needed to establish the process of publishing project's new version releases for public access. I added the new workflow to the same main.workflow file with action Release depended on Artifacts which just does the same release command as "make" but directly with a docker image.

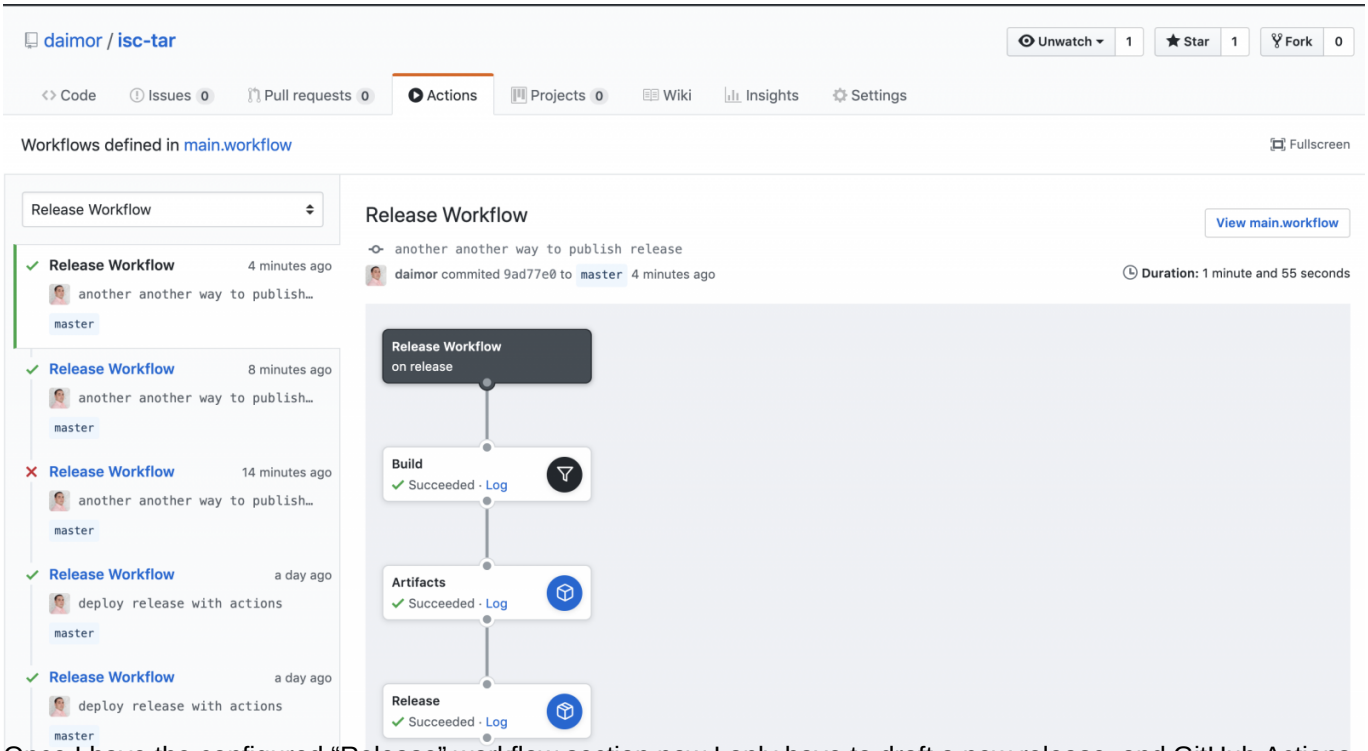
```
workflow "Release Workflow" {
  on = "release"
  resolves = ["Release"]
}

action "Artifacts" {
  needs = ["Build"]
  uses = "docker://daimor/isc-tar"
  runs = "/build_artifacts.sh"
}

action "Release" {
  needs = ["Artifacts"]
  uses = "docker://tsub/ghr"
  secrets = ["GITHUB_TOKEN"]
  runs = "/bin/ash"
  args = ["-c", "ghr -u ${GITHUB_REPOSITORY%/*} -r ${GITHUB_REPOSITORY#*/} ${GITHUB_R
EF##*/} out"]
}
```

To implement "Release" action I found the image on docker hub which helps publishing GitHub releases.BTW it is so cool when you can find any public docker image which does you the job, and then you can use it as an action. But later I found a better solution and did it in this way. Which looks much simpler.

```
action "Release" {
  needs = ["Artifacts"]
  uses = "JasonEtco/upload-to-release@master"
  secrets = ["GITHUB_TOKEN"]
  args = "out/zUtils.FileBinaryTar.xml"
}
```



Once I have the configured “Release” workflow section now I only have to draft a new release, and GitHub Actions will build it and attach there.

You can find all of these sources in my repository. Unfortunately, looks like this actions tab available only for when you signed for the beta.

#Containerization #Continuous Integration #Development Environment #DevOps #Docker #GitHub #InterSystems IRIS
30 0 1 0 317

Source URL: https://community.intersystems.com/post/behind-scene-isc-tar-project-and-story-about-continuous-integration-using-github-actions