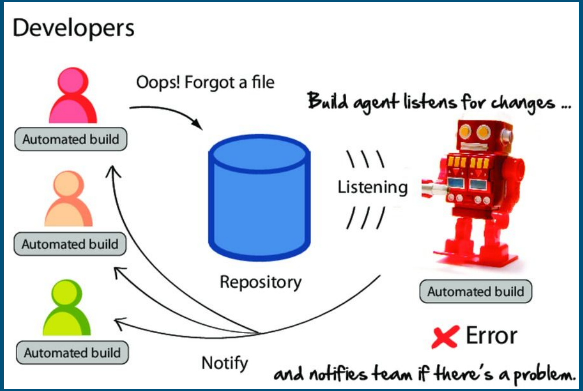
Automated builds

What is an automated build?

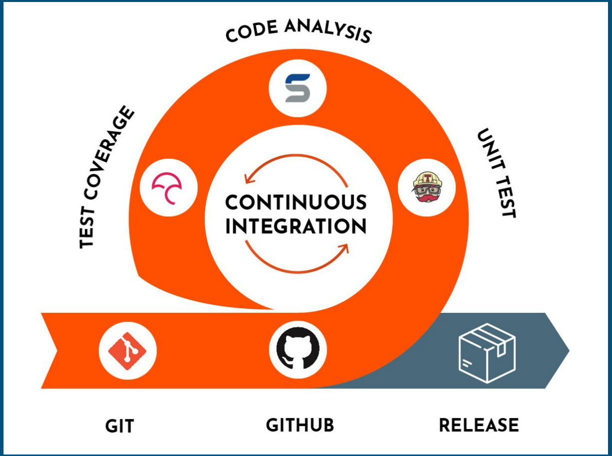


Types of automated builds

- There are 3 types:

Continuous integration (CI)

- The continuous integration is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early.

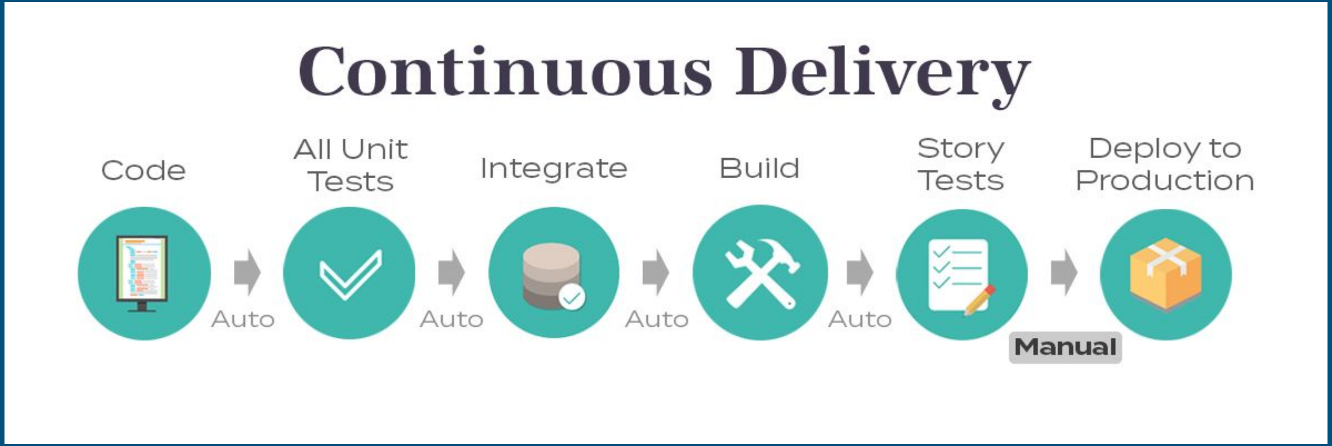


Benefits of continuous integration

- Say goodbye to long and tense integrations
- Increase visibility enabling greater communication
- Catch issues early and nip them in the bud
- Spend less time debugging and more time adding features
- Build a solid foundation
- Stop waiting to find out if your code's going to work
- Reduce integration problems allowing you to deliver software more rapidly

Continuous delivery (CD)

- Continuous Delivery is the ability to get changes of all types, including new features, configuration changes, bug fixes and experiments into production, or into the hands of users, *safely* and *quickly* in a *sustainable* way.



Benefits of continuous delivery

- **Low risk releases**: make the software deployments painless
- **Faster time to market**
- **Higher quality**
- **Lower costs**
- **Better products**
- **Happier team**

Continuous deployment (CD)

Continuous deployment is a strategy for software releases where in any code commit that passes the automated testing phase is automatically released into the production environment, making changes that are visible to the software's users.

Benefits of continuous deployment

- Eliminate DIY for Continuous Delivery and increase the focus on the product.
- Automate the repetitive tasks and focus on actual testing.
- Make deployments frictionless without compromising security.
- Scale from a single application to an Enterprise IT portfolio.
- Connect your existing tools and technologies (such as CI providers, DevOps tools, or scripts) into a harmonious workflow.
- Integrate teams and processes with a unified pipeline.
- Create workflows across the development, testing, and production environments
- Provide a single view across all applications and environments.
- Ship both cloud-native and traditional applications in a unified pipeline.
- Improve overall productivity.

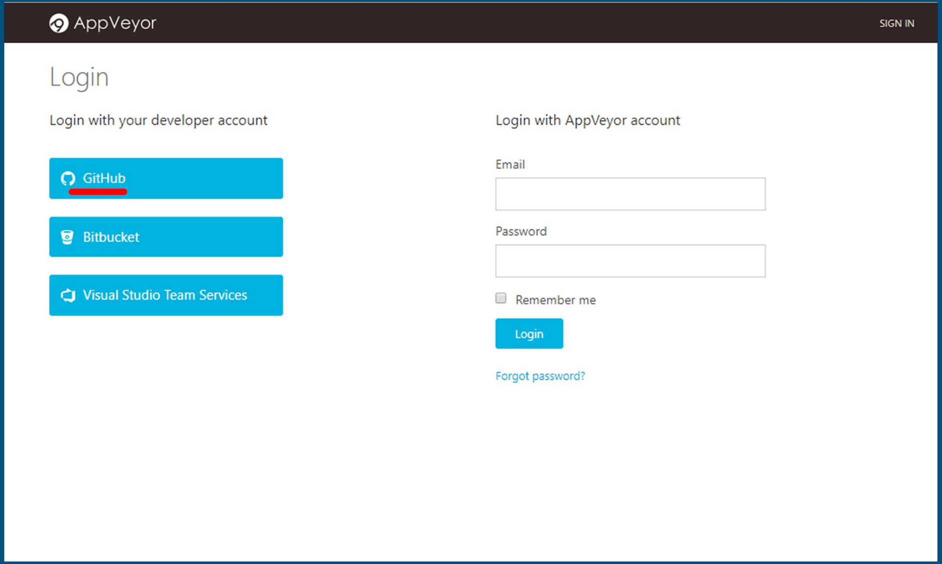
AppVeyor

What is AppVeyor

AppVeyor is a program that every time a commit is done to the code, it automatically uploads the build with all the needed artifacts to the Release page of GitHub giving you feedback of how the build has been done.

Tutorial

Starting


AppVeyor

SIGN IN

Login

Login with your developer account

GitHub

Bitbucket

Visual Studio Team Services

Login with AppVeyor account

Email

Password

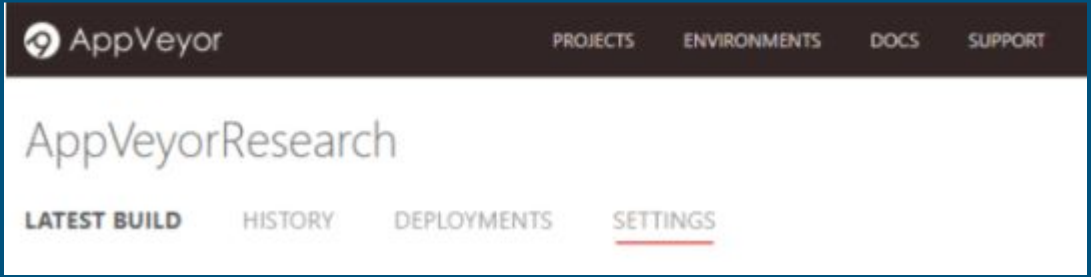
Remember me

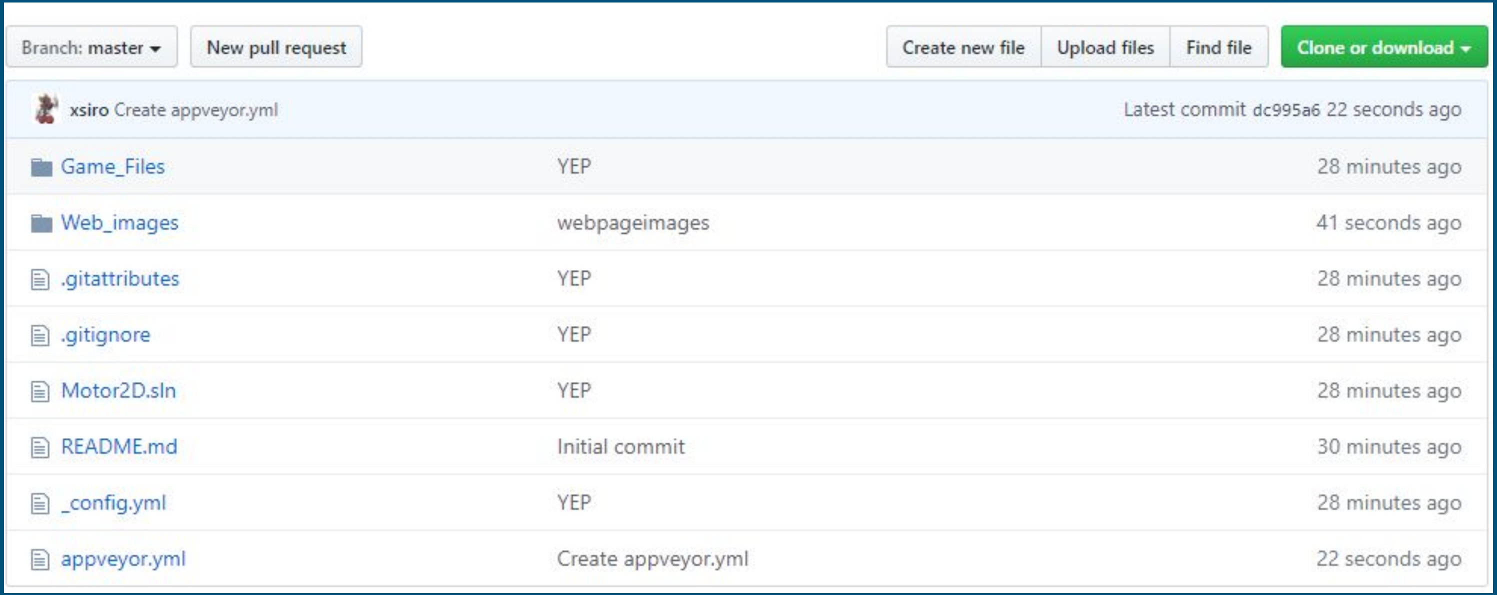
Login

Forgot password?

- Once we have synchronized both applications, we can go on with AppVeyor creating a new project and selecting the GitHub repository which we want to have automated builds.
- Now we have our project in AppVeyor, by default every time we make a commit, it will try to make a built, but it probably fails due to the app configuration is not the correct. So the next step is how to configure it.

Configuration





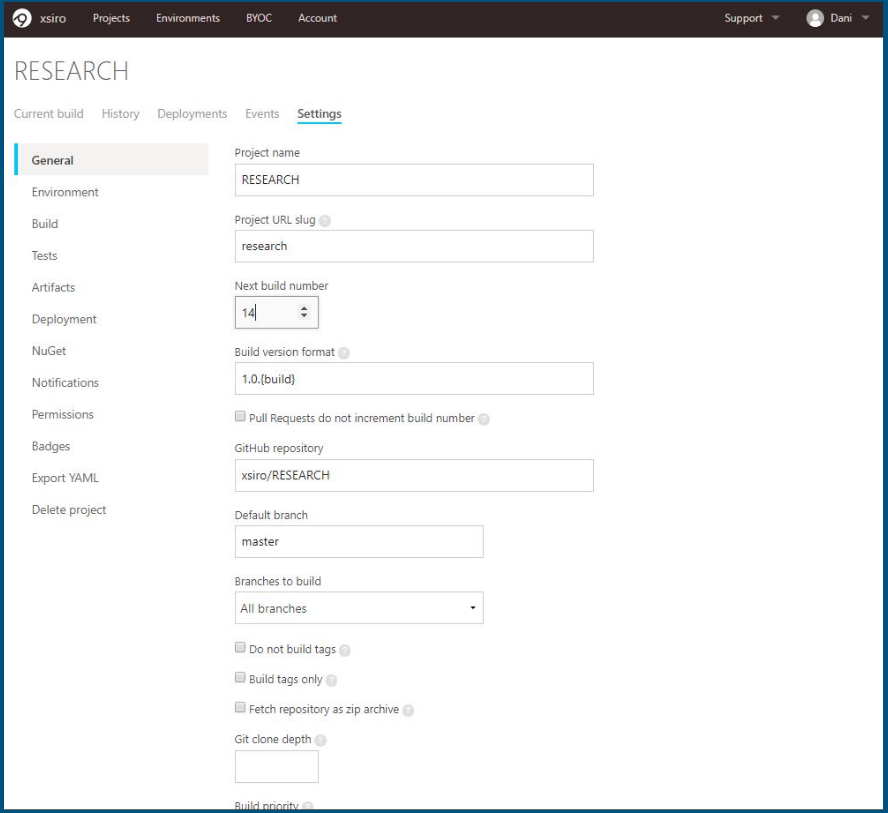
Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

xsiro Create appveyor.yml — Latest commit dc995a6 22 seconds ago

Game_Files	YEP	28 minutes ago
Web_images	webpageimages	41 seconds ago
.gitattributes	YEP	28 minutes ago
.gitignore	YEP	28 minutes ago
Motor2D.sln	YEP	28 minutes ago
README.md	Initial commit	30 minutes ago
_config.yml	YEP	28 minutes ago
appveyor.yml	Create appveyor.yml	22 seconds ago

It’s needed to remark that **AppVeyor will give preference to the YAML file before the project settings**. So be careful.

The project settings is divided in different sections, the main one is *General*. There, the most relevant option is that you can configure the *Build version format*, that will increase every time a built is done (regardless of if it fails). Another useful setting is that you can select from which branch you want to make the built every time a commit is done, in *Default branch* and *Branches to build*.



RESEARCH

Project name: RESEARCH

Project URL slug: research

Next build number: 14

Build version format: 1.0.{build}

[ ] Pull Requests do not increment build number

GitHub repository: xsiro/RESEARCH

Default branch: master

Branches to build: All branches

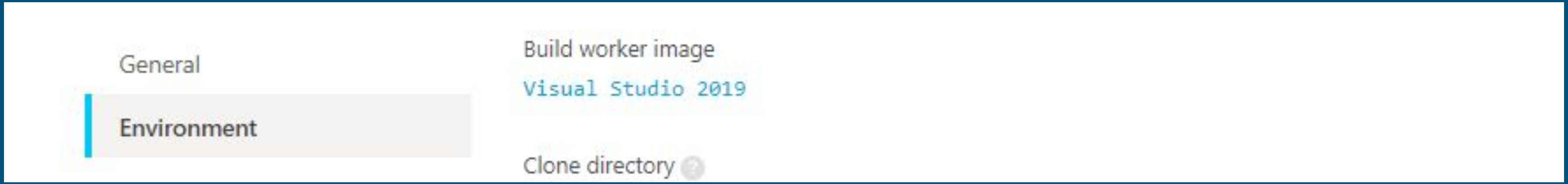
[ ] Do not build tags

[ ] Build tags only

[ ] Fetch repository as zip archive

Git clone depth:

Build priority

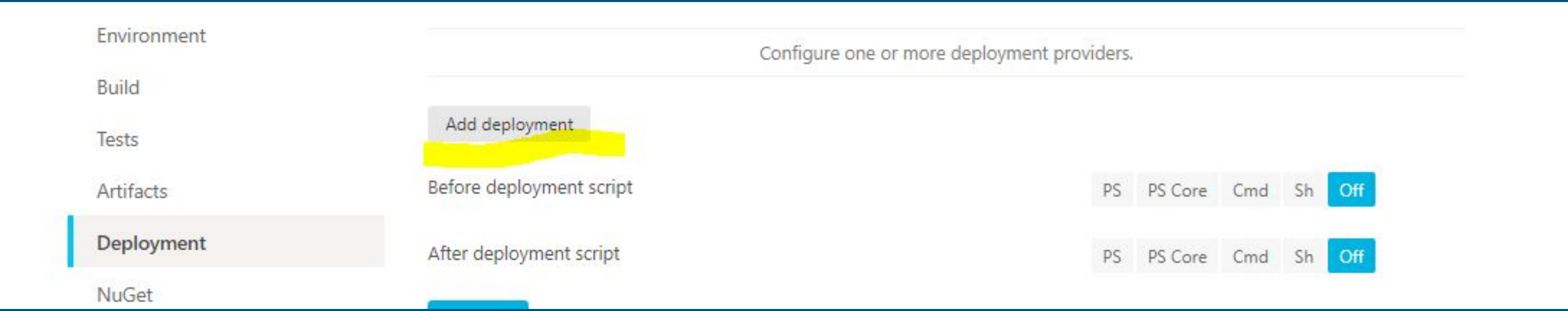

General

Environment

Build worker image

Visual Studio 2019

Clone directory

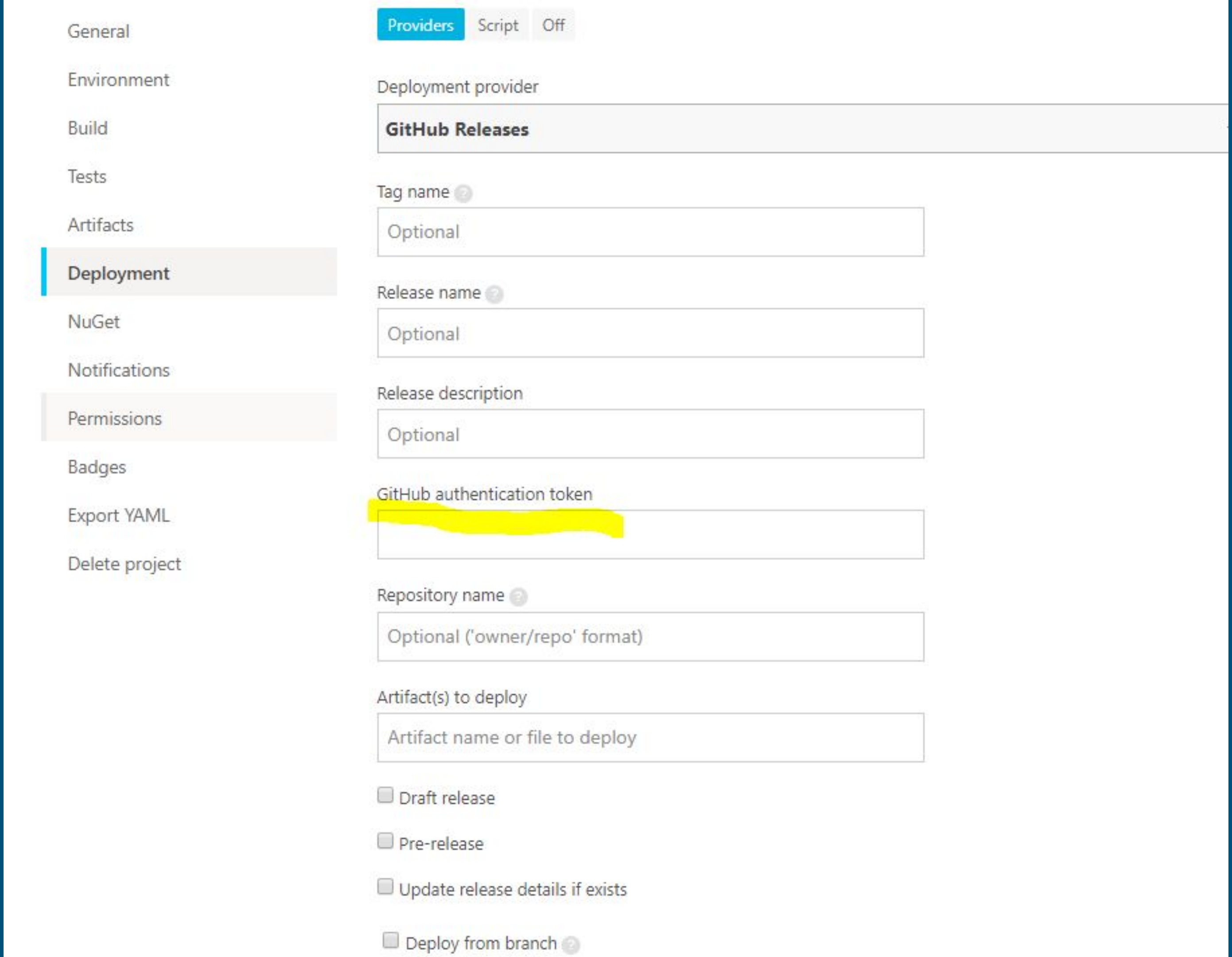

Environment
Build
Tests
Artifacts
Deployment
NuGet

Configure one or more deployment providers.

Add deployment

Before deployment script — PS | PS Core | Cmd | Sh | [✓ Off]

After deployment script — PS | PS Core | Cmd | Sh | [✓ Off]



Deployment provider: **GitHub Releases**

Tag name: Optional

Release name: Optional

Release description: Optional

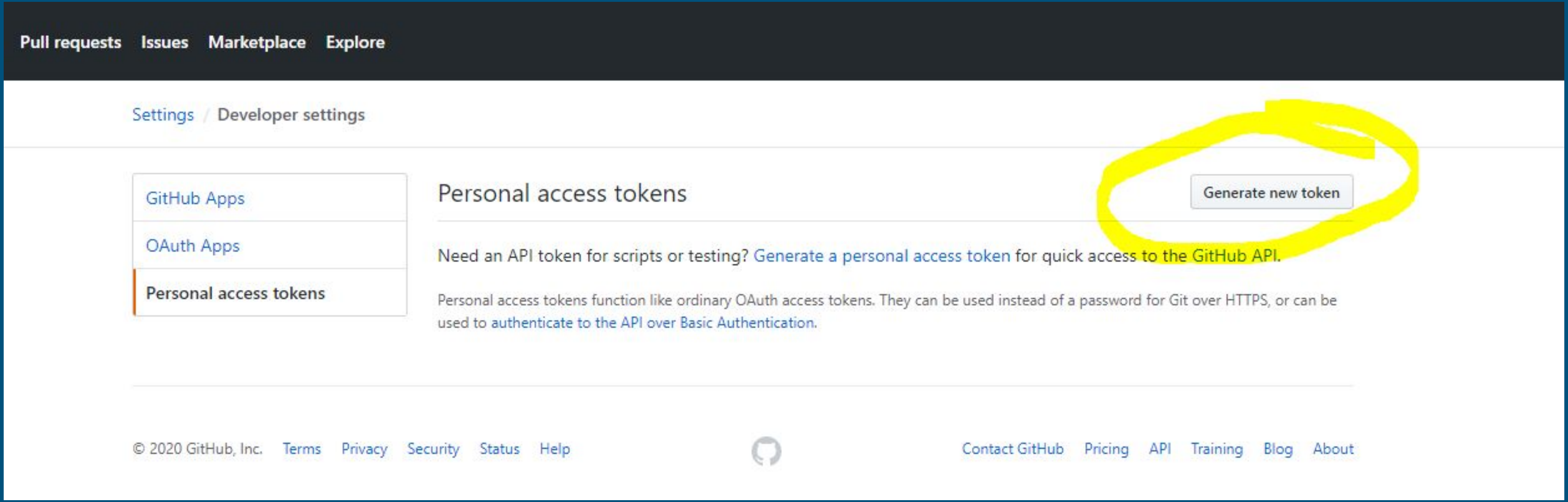
GitHub authentication token:

Repository name: Optional ('owner/repo' format)

Artifact(s) to deploy: Artifact name or file to deploy

- [ ] Draft release
- [ ] Pre-release
- [ ] Update release details if exists
- [ ] Deploy from branch

How to get GitHub authentication token



Pull requests Issues Marketplace Explore

Settings / Developer settings

GitHub Apps
OAuth Apps
Personal access tokens

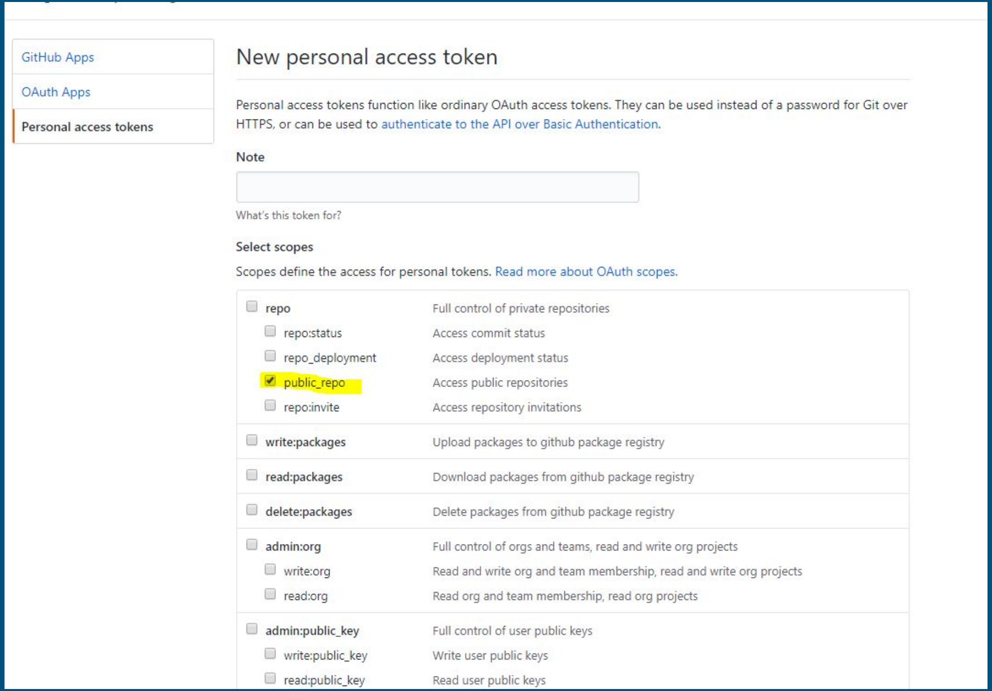
Personal access tokens

Generate new token

Need an API token for scripts or testing? Generate a personal access token for quick access to the GitHub API.

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

© 2020 GitHub, Inc. Terms Privacy Security Status Help Contact GitHub Pricing API Training Blog About


GitHub Apps
OAuth Apps
Personal access tokens

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

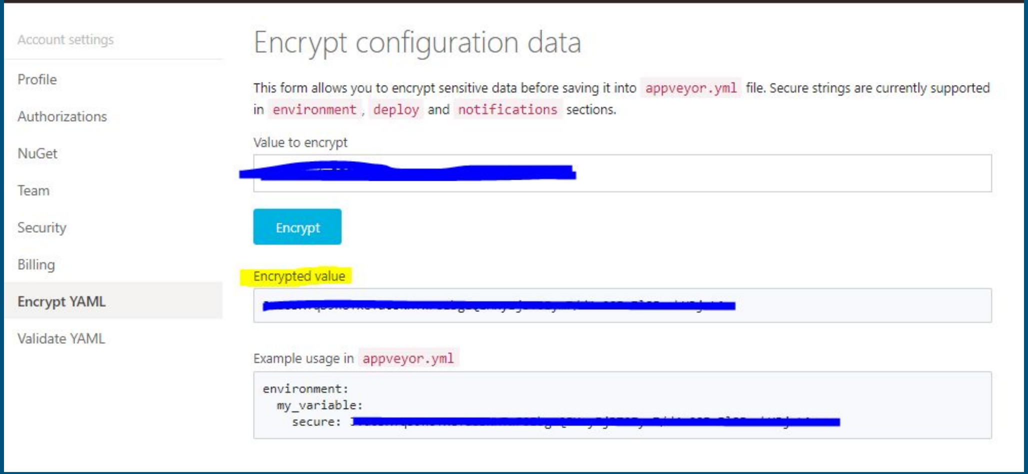
Note

What's this token for?

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

[ ] repo	Full control of private repositories
[ ] repo:status	Access commit status
[ ] repo_deployment	Access deployment status
[✓] public_repo	Access public repositories
[ ] repo:invite	Access repository invitations
[ ] write:packages	Upload packages to github package registry
[ ] read:packages	Download packages from github package registry
[ ] delete:packages	Delete packages from github package registry
[ ] admin:org	Full control of orgs and teams, read and write org projects
[ ] write:org	Read and write org and team membership, read and write org projects
[ ] read:org	Read org and team membership, read org projects
[ ] admin:public_key	Full control of user public keys
[ ] write:public_key	Write user public keys
[ ] read:public_key	Read user public keys



Encrypt configuration data

This form allows you to encrypt sensitive data before saving it into `appveyor.yml` file. Secure strings are currently supported in `environment`, `deploy` and `notifications` sections.

Value to encrypt

[redacted]

Encrypted value

[redacted]

Example usage in `appveyor.yml`

```
environment:
  my_variable:
    secure: [redacted]
```

Back to AppVeyor

At this point AppVeyor is capable to access to the Release GitHub page.

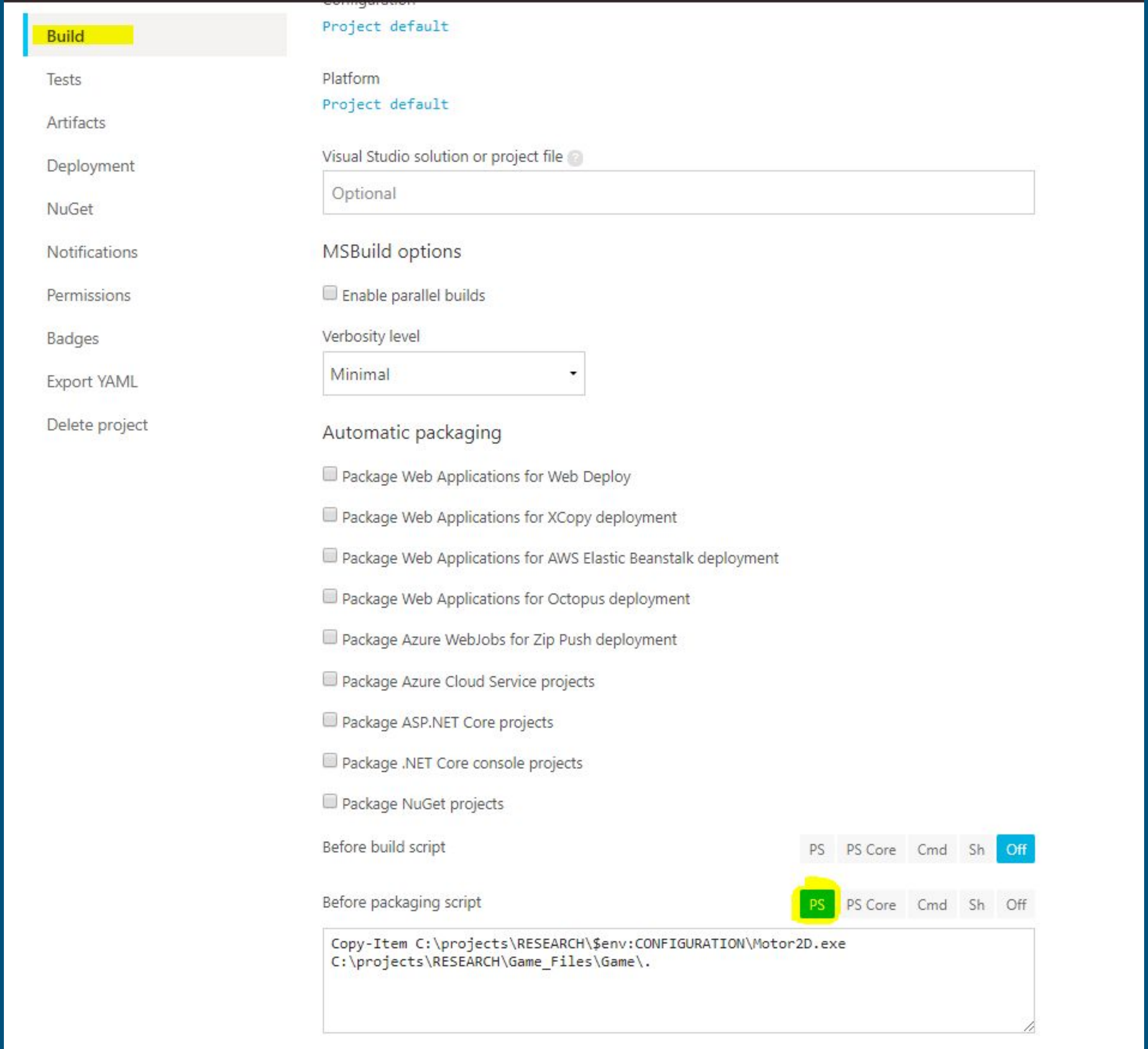
So our objective is to make AppVeyor do automated builds from our GitHub repository, but we need to remind which items a build should have:

- A README.md file
- A folder with all the Assets of the game and the libraries .dll
- The executable of the game .exe

It is recommended putting together in a folder the ReadMe, the assets and the libraries to make the process easily. In all the explanation we will refer to this folder as *\Game*.

The script is the following:

```
Copy-Item C:\projects\(your_project_name)\$env:CONFIGURATION\(your_solution_name).exe
C:\projects\(your_project_name)\Game\.
```



Project default

Platform
Project default

Visual Studio solution or project file
Optional

MSBuild options

☐ Enable parallel builds

Verbosity level
Minimal

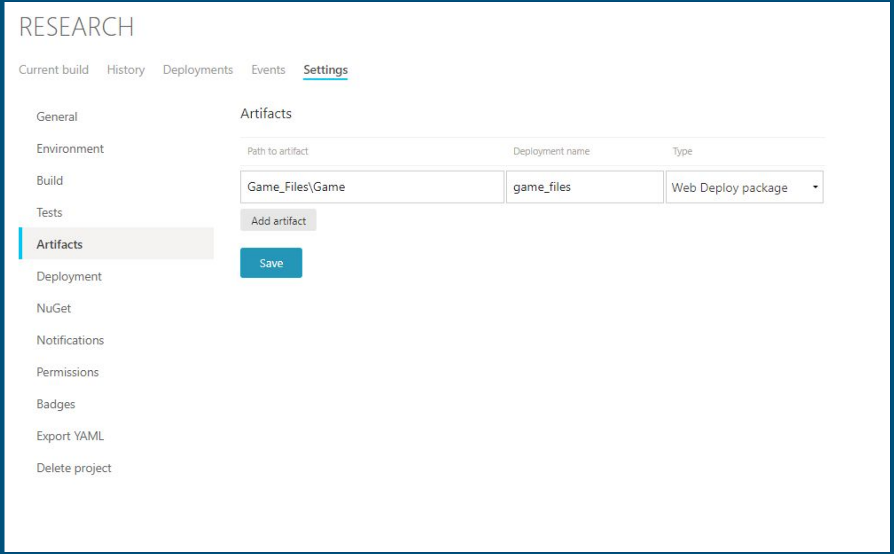
Automatic packaging

☐ Package Web Applications for Web Deploy
☐ Package Web Applications for XCopy deployment
☐ Package Web Applications for AWS Elastic Beanstalk deployment
☐ Package Web Applications for Octopus deployment
☐ Package Azure WebJobs for Zip Push deployment
☐ Package Azure Cloud Service projects
☐ Package ASP.NET Core projects
☐ Package .NET Core console projects
☐ Package NuGet projects

Before build script: PS PS Core Cmd Sh [✓ Off]

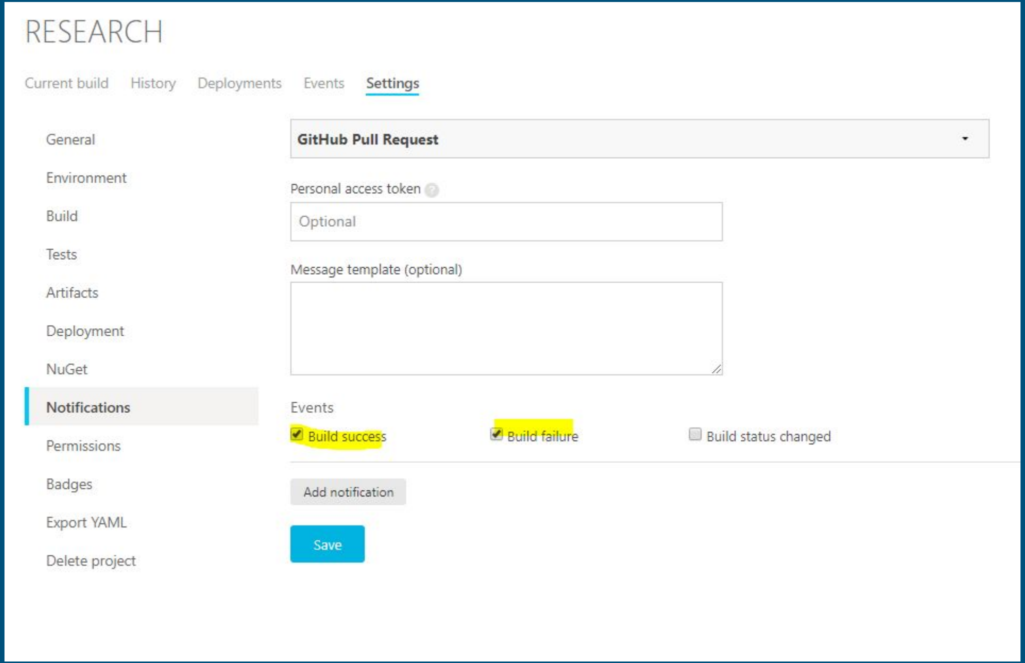
Before packaging script: [✓ PS] PS Core Cmd Sh Off

```
Copy-Item C:\projects\RESEARCH\$env:CONFIGURATION\Motor2D.exe
C:\projects\RESEARCH\Game_Files\Game\.
```



Artifacts

Path to artifact	Deployment name	Type
Game_Files\Game	game_files	Web Deploy package


RESEARCH

Current build · History · Deployments · Events · **Settings**

General
Environment
Build
Tests
Artifacts
Deployment
NuGet
Notifications
Permissions
Badges
Export YAML
Delete project

GitHub Pull Request

Personal access token

Optional

Message template (optional)

Events

[✓] Build success [✓] Build failure [ ] Build status changed

Add notification

Save