

Hand­s-on lab

Lab: CI e CD com Jenkins no Azure

Abril/2016, versão 1.0

Autor: Allan Targino, Evangelista Técnico



Conteúdo

[1) Introdução 3](#_Toc479378185)

[2) Pré-Requisitos 3](#_Toc479378186)

[3) Criação da VM no Azure e Instalação do Jenkins 3](#_Toc479378187)

[a) Windows 4](#_Toc479378188)

[b) Linux 9](#_Toc479378189)

[4) Instalando plugins adicionais 13](#_Toc479378190)

[5) Continuous Integration 16](#_Toc479378191)

[6) Continuous Delivery 27](#_Toc479378192)

[7) Guardando artefatos com alta disponibilidade no Azure Blob Storage 30](#_Toc479378193)

[8) Adicionando a segurança do Azure AD ao nosso servidor 31](#_Toc479378194)

[9) Integração e orquestração de Builds através do VSTS 31](#_Toc479378195)

[10) Usando do poder de extensibilidade do Jenkins e do Azure 31](#_Toc479378196)

[a) Usando a REST API do Jenkins para projetos ou integrações personalizadas 31](#_Toc479378197)

[b) Usando as REST APIs do Azure com etapas de Build/Release do Jenkins 31](#_Toc479378198)

[11) Referências 32](#_Toc479378199)

# Introdução

Jenkins é uma ferramenta Open-Source do mundo de DevOps bastante famosa, o qual originalmente se chamava Hudson. Para uma pequena introdução do que exatamente Jenkins faz, extrai um pequeno trecho do livro “Jenkins: The Definitive Guide”, de John Ferguson Smart, O'Reilly Media – 2011:

“Boasting a dominant market share, Jenkins is used by teams of all sizes, for projects in a wide variety of languages and technologies, including .NET, Ruby, Groovy, Grails, PHP and more, as well as Java. So what has made Jenkins such a success? And why use Jenkins for your CI infrastructure?

Firstly, Jenkins is easy to use. The user interface is simple, intuitive, and visually appealing, and Jenkins as a whole has a very low learning curve. As we will see in the next chapter, you can get started with Jenkins in a matter of minutes. However Jenkins does not sacrifice power or extensibility: it is also extremely flexible and easy to adapt to your own purposes. Hundreds of open source plugins are available, with more coming out every week. These plugins cover everything from version control systems, build tools, code quality metrics, build notifiers, integration with external systems, UI customization, games, and much more. And installing them is quick and easy. […]”

Neste Hands-On-Lab (HOL) veremos um cenário fim a fim de Continuous integration (CI) e Continuous deployment (CD) com Jenkins, mostrando como o Azure pode ser um fator determinante na construção e deploy de aplicações. Será mostrado também como o Visual Studio Team Services (VSTS) pode atuar em conjunto com este cenário de Jenkins.

# Pré-Requisitos

Para este HOL serão necessários:

1. Conta de Azure
   * Caso não possua, consiga uma conta trial com R$ 750,00 [neste link](https://azure.microsoft.com/pt-br/free/).
2. Conta no Github
   * Caso não possua, crie uma gratuitamente [aqui](https://github.com/join).

Apesar de opcionais, é interessante possuir instalado:

1. Algum software para fazer chamadas REST (Ex: [Postman](https://www.getpostman.com/))

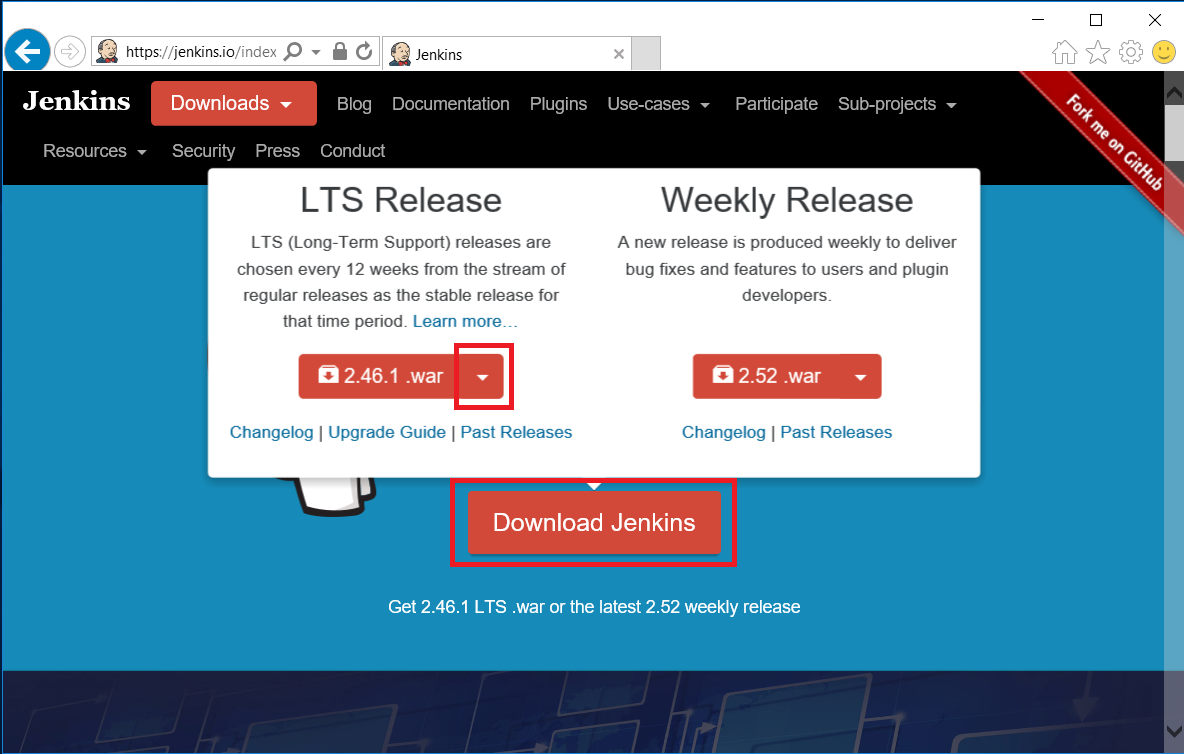
# Criação da VM no Azure e Instalação do Jenkins

Podemos instalar o Jenkins no Linux ou Windows. Você deve escolher o sistema operacional essencialmente baseado na necessidade de SO de seu build. Por exemplo, para fazer o build de uma aplicação .NET Framework (full), é necessária uma máquina Windows com MSBuild instalado. Já para fazer o build de uma aplicação Java, poderíamos usar o Apache Ant em Linux ou Windows. Abaixo listaremos uma maneira de como instalar o Jenkins no SO escolhido em VMs do Azure.

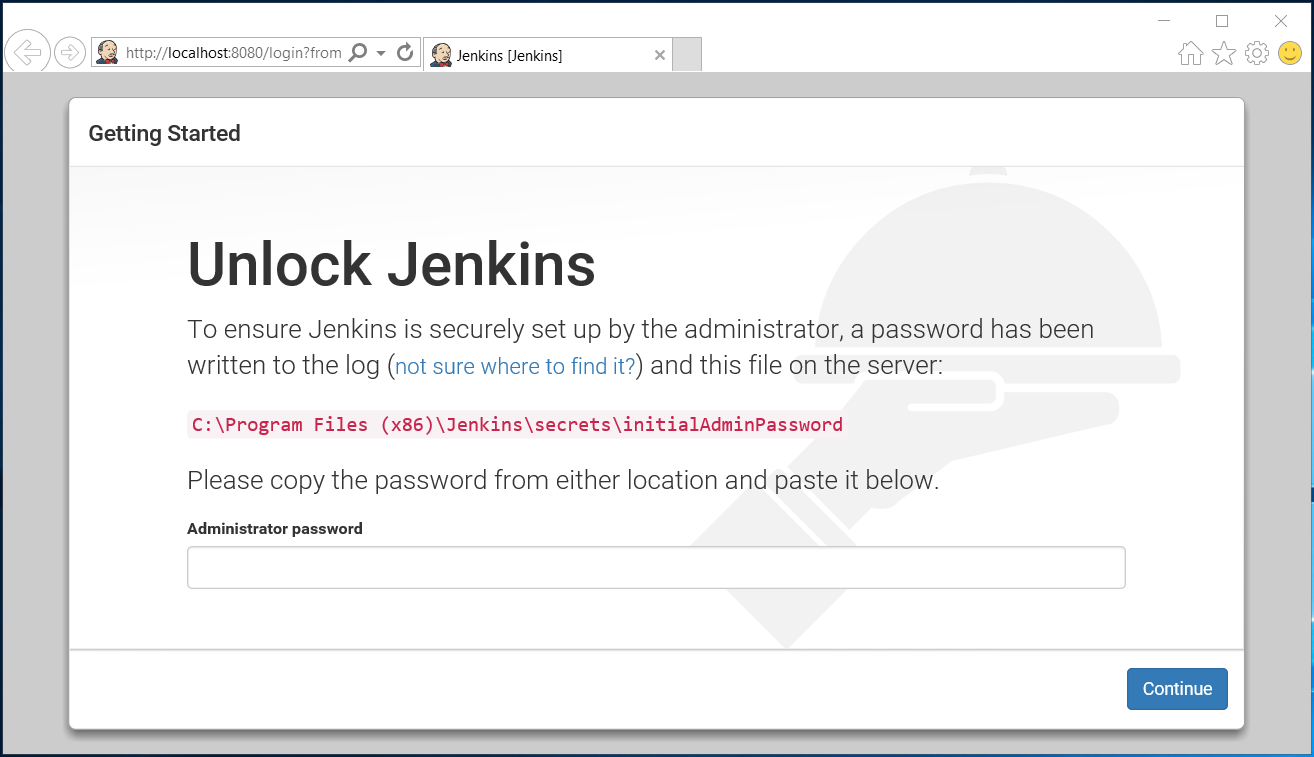
Há varias maneiras de se instalar o Jenkins. Para outras opções ou para consultar um guia mais completo, visite: <https://jenkins.io/download/>

## Windows

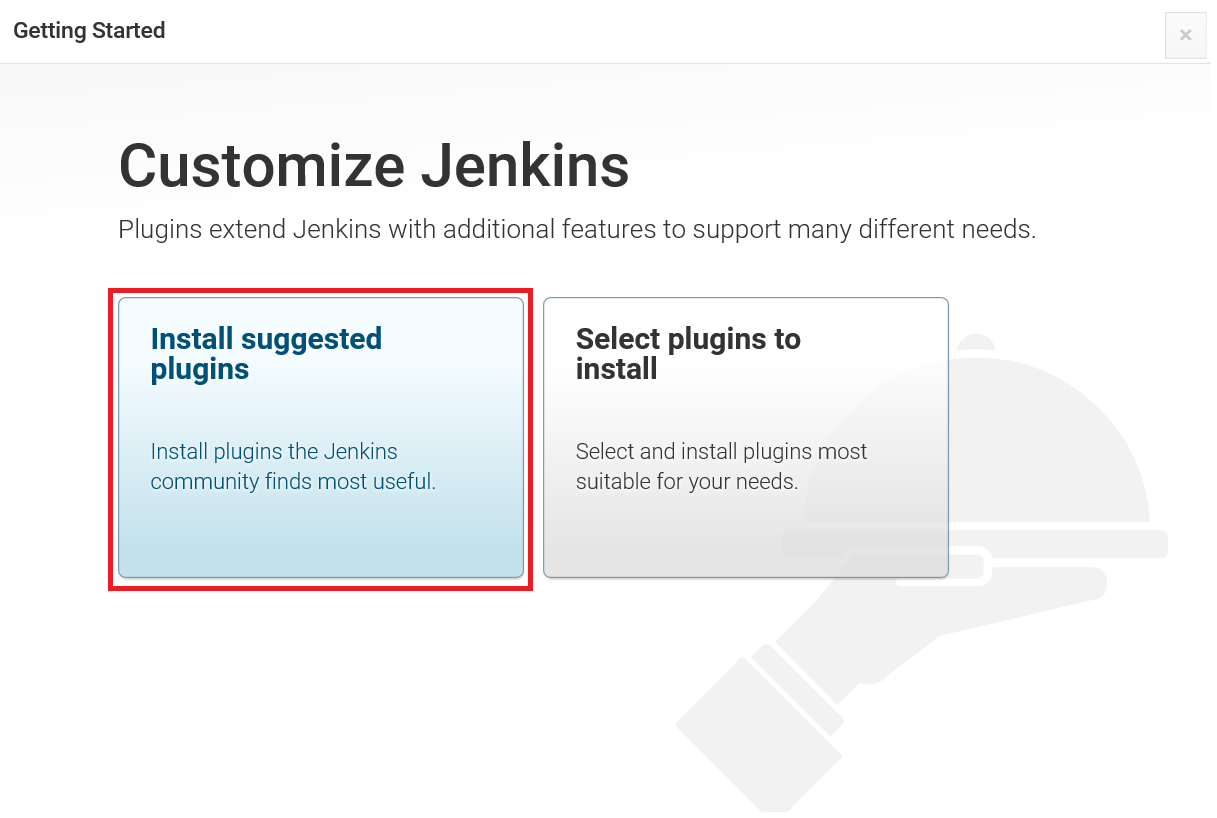
1. Acesse <https://portal.azure.com>
2. Crie uma VM Windows em seu portal. (Passo-a-passo disponível [aqui](https://docs.microsoft.com/en-us/azure/virtual-machines/virtual-machines-windows-quick-create-portal))
3. Após acessar sua VM via RDP, navegue até <https://jenkins.io/>. Clique em “Download Jenkins”, clique no dropdown e por fim clique em Windows. Um download de um arquivo .zip contendo um instalador .msi será iniciado.

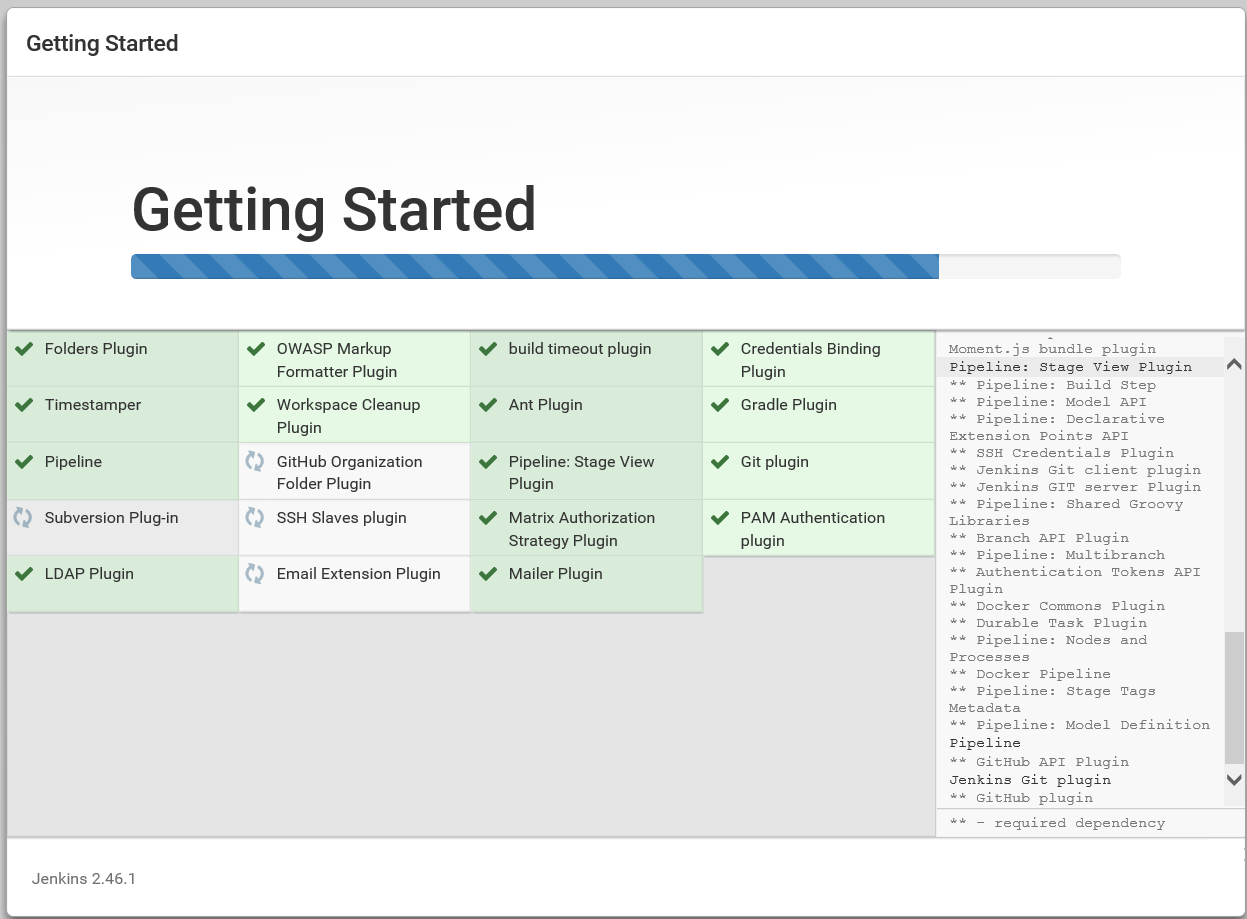


1. A instalação é bem simples: extraia e execute o arquivo .msi. Escolha suas preferências de diretório e avance até a instalação finalizar.
2. Ao finalizar a instalação, se tudo deu certo, o serviço do Jenkins será iniciado na porta 8080 e o browser será aberto em <http://localhost:8080/>.

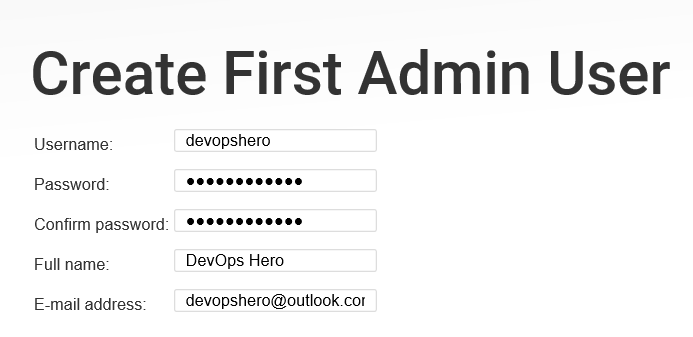


1. Por questões de segurança, você deve copiar uma senha disponível em um arquivo localizado em Diretório de Instalação\secrets\ com o nome de initialAdminPassword e colar dentro da caixa *Administrator password*, clicando em seguida em *Continue*.
2. Ao ser questionado sobre os plugins, escolha a instalação padrão de plugins. Como essa escolha inicial de plugins é limitada, iremos adicionar os outros que iremos precisar no próximo capitulo deste tutorial. Aguarde a instalação dos plugins.

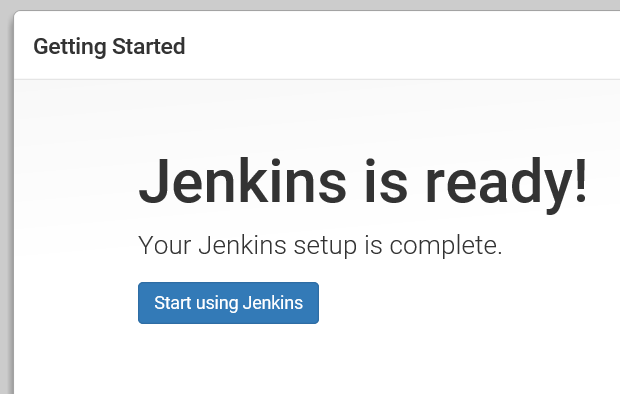


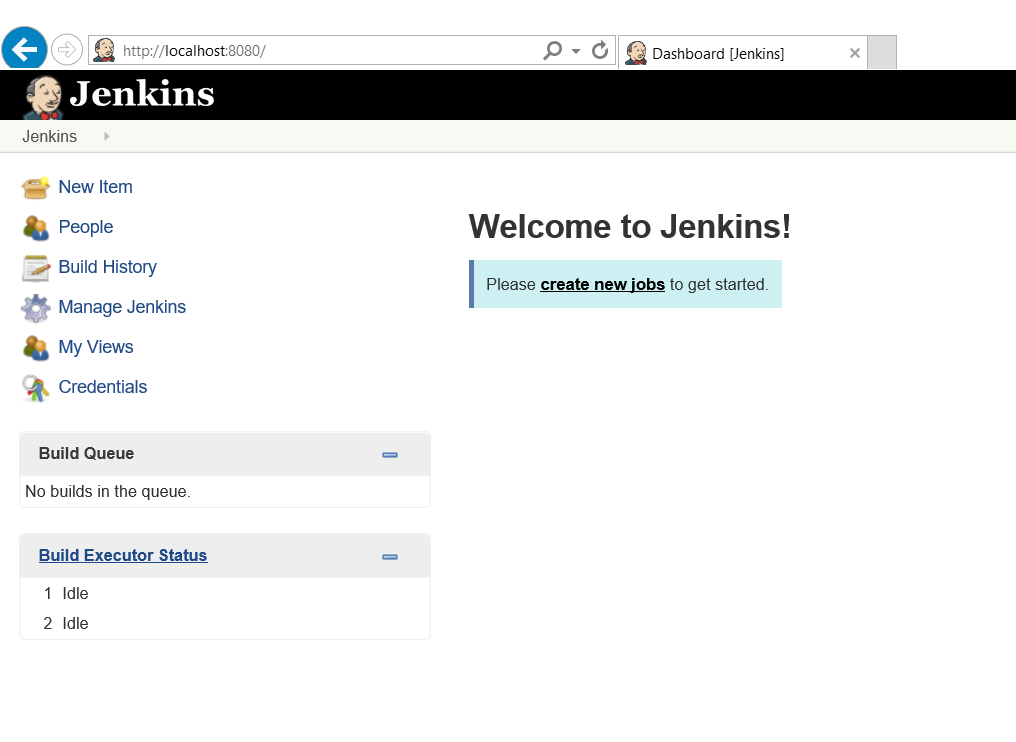


1. Após a instalação bem sucedida dos plugins padrão, entre com os dados do admin local da instância Jenkins e clique no botão “*Save and Finish*”. Em uma etapa posterior iremos usar usuários de um Azure AD, adicionando mais segurança a nossa instalação do Jenkins.

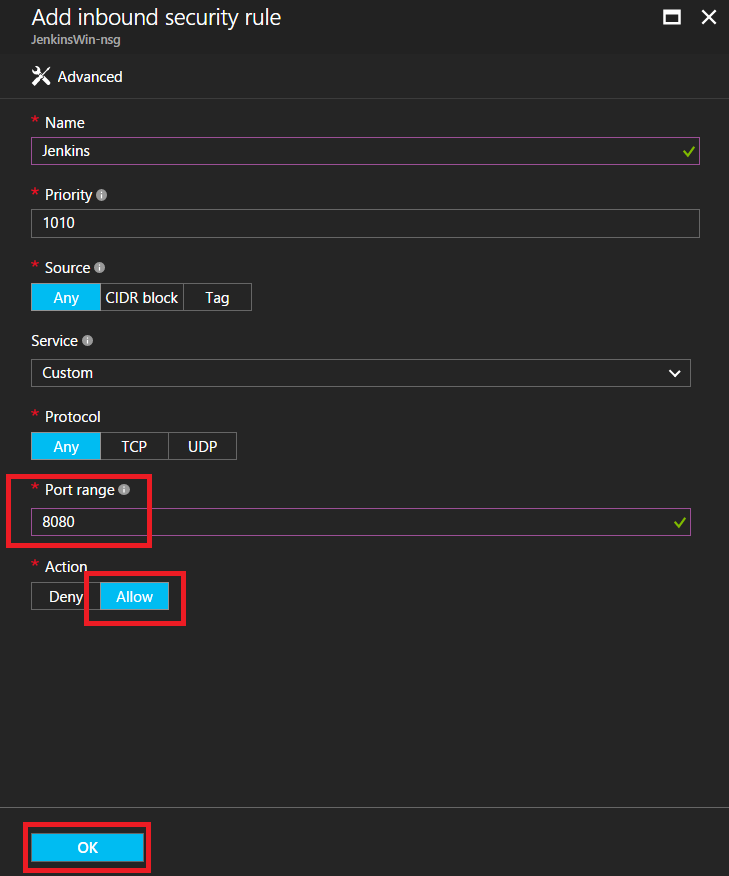


1. Pronto! Sua instalação foi concluída com sucesso no Windows!





1. Opcional: Iremos abrir as portas do firewall no Azure para que possamos acessar nossa instância do Jenkins por um browser externo. Feche a conexão RDP e vá até o Network Security Group (NSG) associado a sua VM e permite a regra de entrada para a porta 8080. Para detalhes do processo um guia de introdução aos NSG pode ser encontrado [aqui](https://docs.microsoft.com/pt-br/azure/virtual-machines/windows/nsg-quickstart-portal).



1. Abra um browser em sua máquina local e digite o IP público de sua VM e a porta 8080, se o processo deu certo, a instância Jenkins deve ser navegável em sua máquina.



## Linux

1. Acesse <https://portal.azure.com>
2. Crie uma VM Linux em seu portal. (Passo-a-passo disponível [aqui](https://docs.microsoft.com/pt-br/azure/virtual-machines/linux/quick-create-portal))
3. Após acessa-la via SSH, entre com os seguintes comandos para instalar o Jenkins

$ wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -

$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

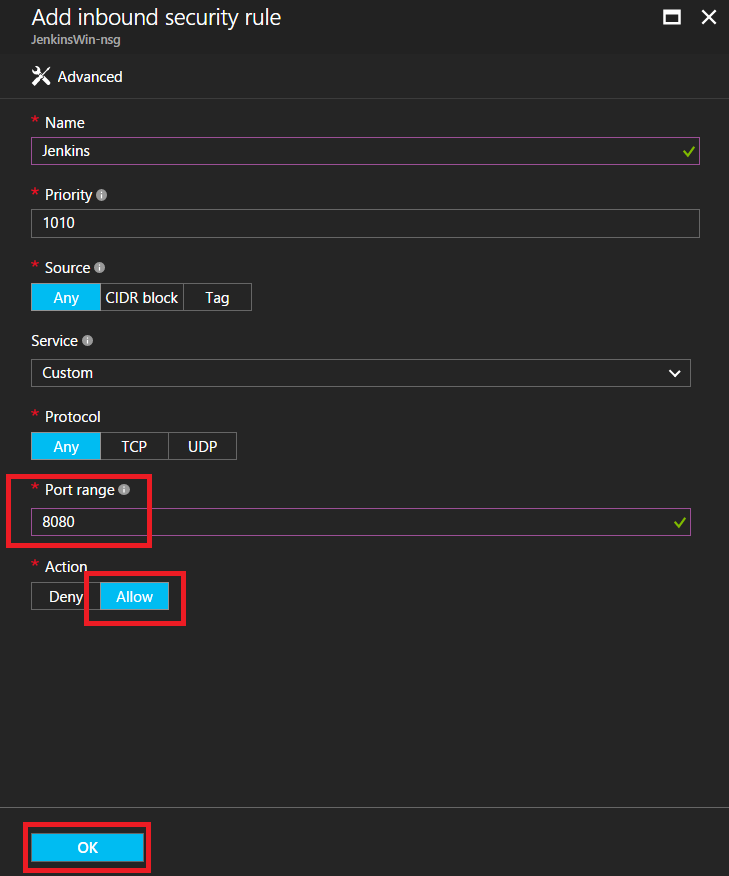
$ sudo apt-get update

$ sudo apt-get install Jenkins

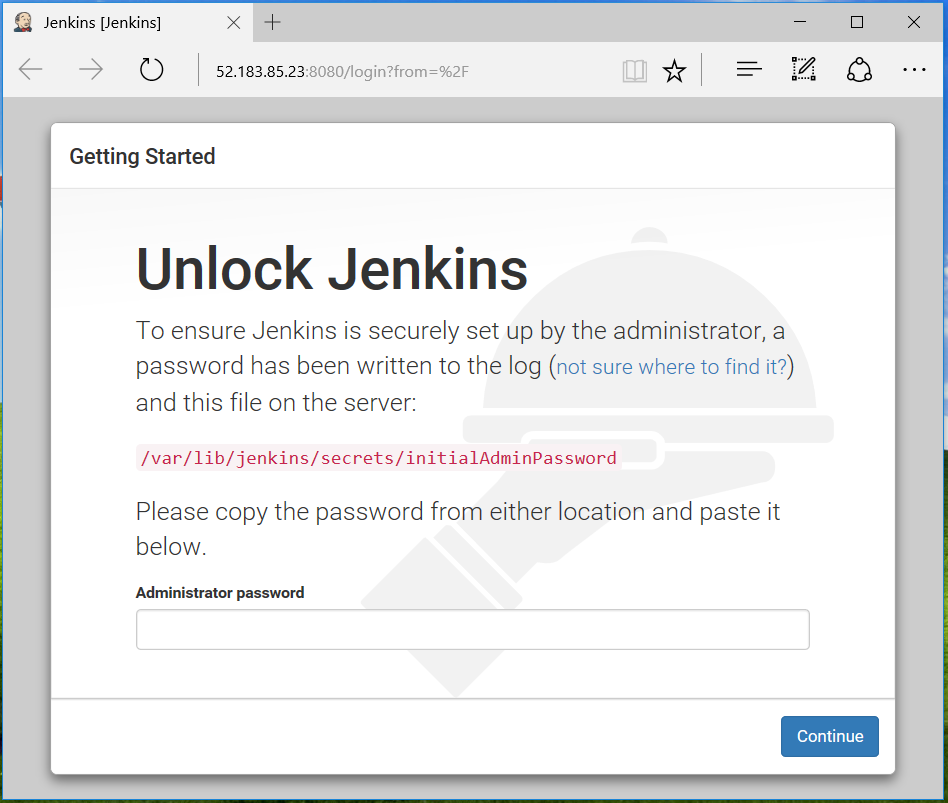
Veja se o serviço está rodando ao dar o commando:

$ service --status-all

1. Iremos abrir as portas do firewall no Azure para que possamos acessar nossa instância do Jenkins por um browser externo. Feche a conexão RDP e vá até o Network Security Group (NSG) associado a sua VM e permite a regra de entrada para a porta 8080. Para detalhes do processo um guia de introdução aos NSG pode ser encontrado [aqui](https://docs.microsoft.com/pt-br/azure/virtual-machines/windows/nsg-quickstart-portal).



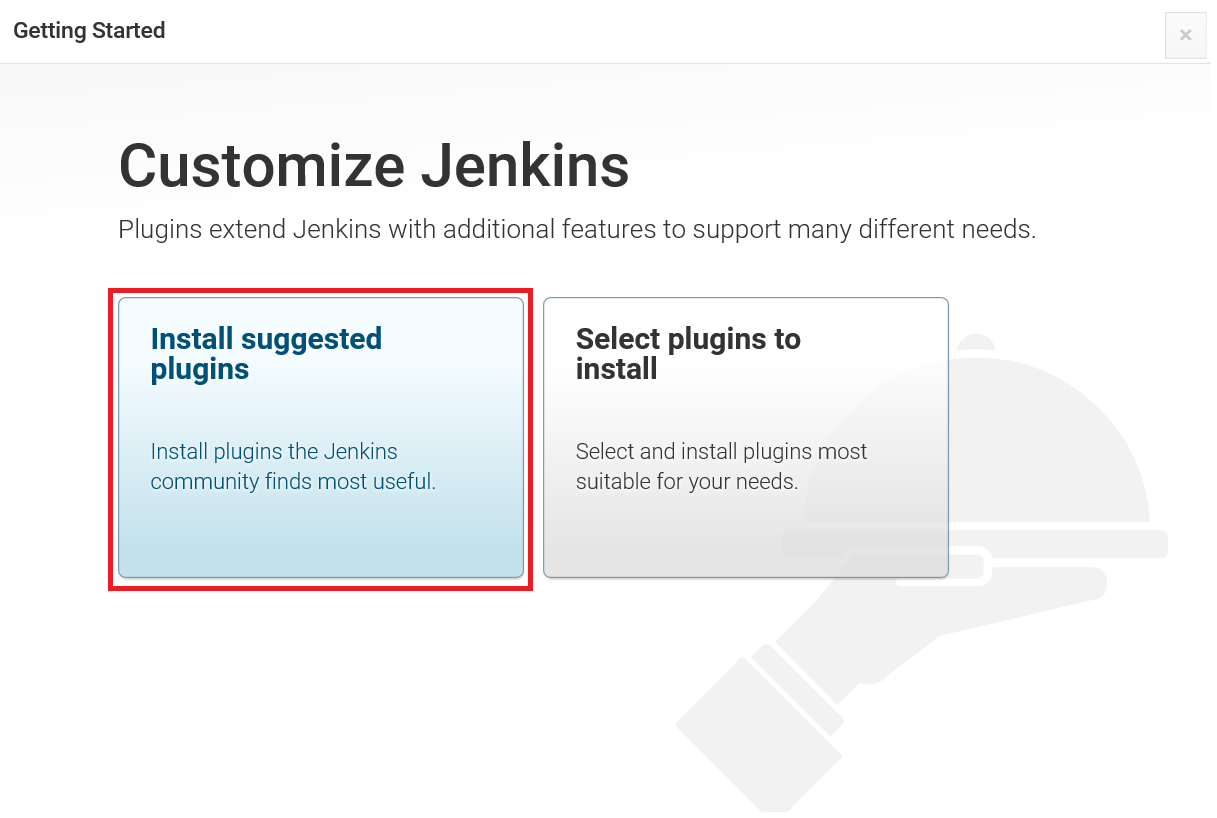
1. Abra um browser em sua máquina local e digite o IP público de sua VM e a porta 8080, se o processo deu certo, a instância Jenkins deve ser navegável em sua máquina.

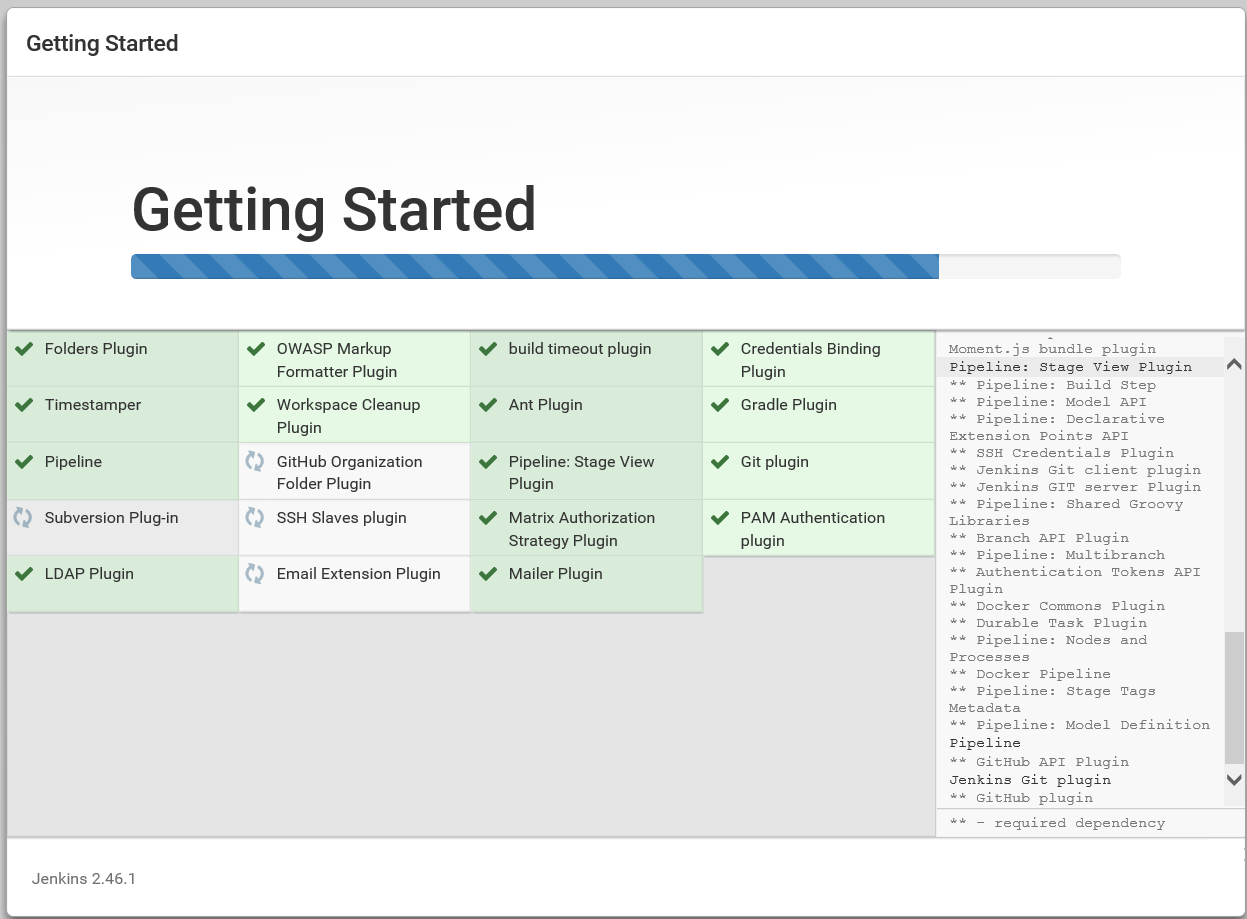


1. Por questões de segurança, você deve copiar uma senha disponível em um arquivo localizado em Diretório de Instalação\secrets\ com o nome de initialAdminPassword e colar dentro da caixa *Administrator password*, clicando em seguida em *Continue*.

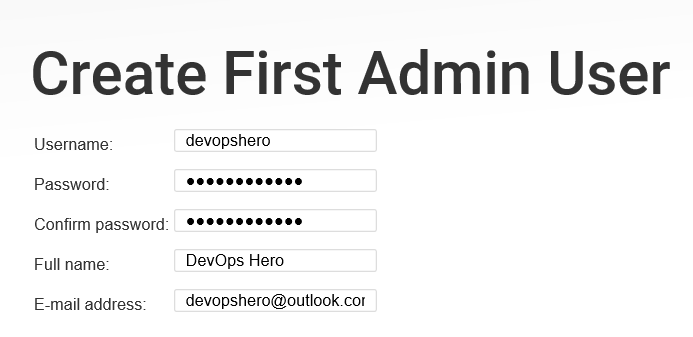
$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword

1. Ao ser questionado sobre os plugins, escolha a instalação padrão de plugins. Como essa escolha inicial de plugins é limitada, iremos adicionar os outros que iremos precisar no próximo capitulo deste tutorial. Aguarde a instalação dos plugins.

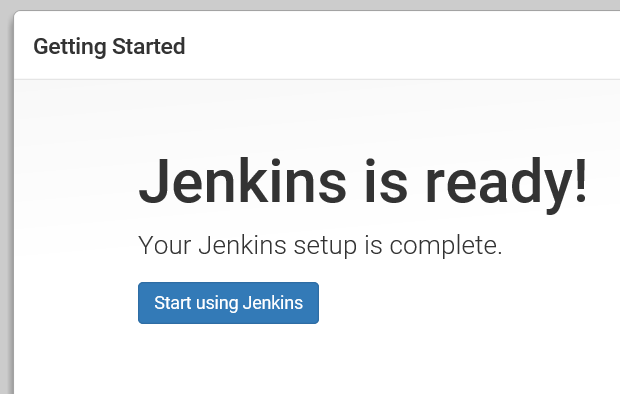


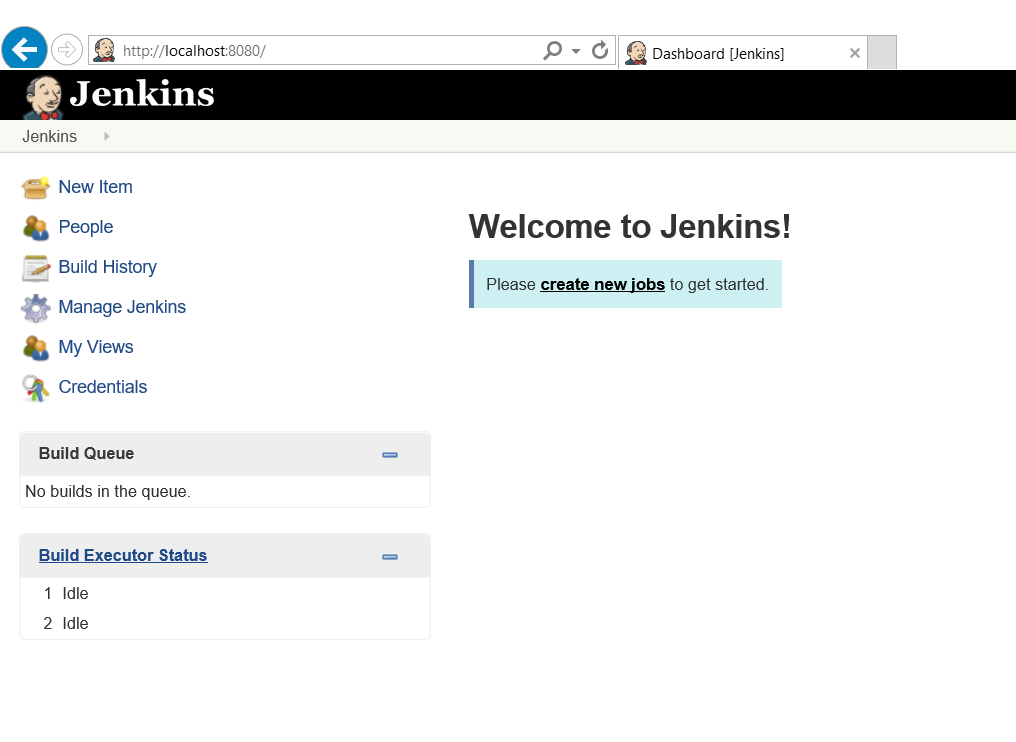


1. Após a instalação bem sucedida dos plugins padrão, entre com os dados do admin local da instância Jenkins e clique no botão “*Save and Finish*”. Em uma etapa posterior iremos usar usuários de um Azure AD, adicionando mais segurança a nossa instalação do Jenkins.



1. Pronto! Sua instalação foi concluída com sucesso no Linux!

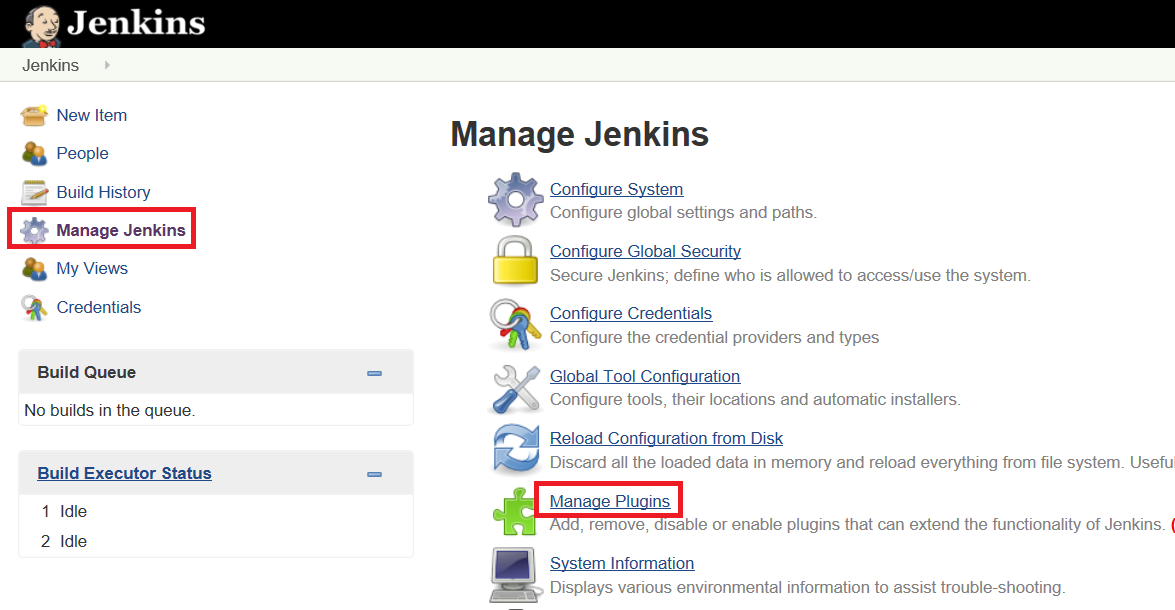




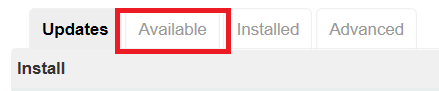
# Instalando plugins adicionais

Definitivamente um dos pontos mais fortes do Jenkins é a extensão de suas funcionalidades através de plugins. Hoje já plugins para absolutamente quase tudo.

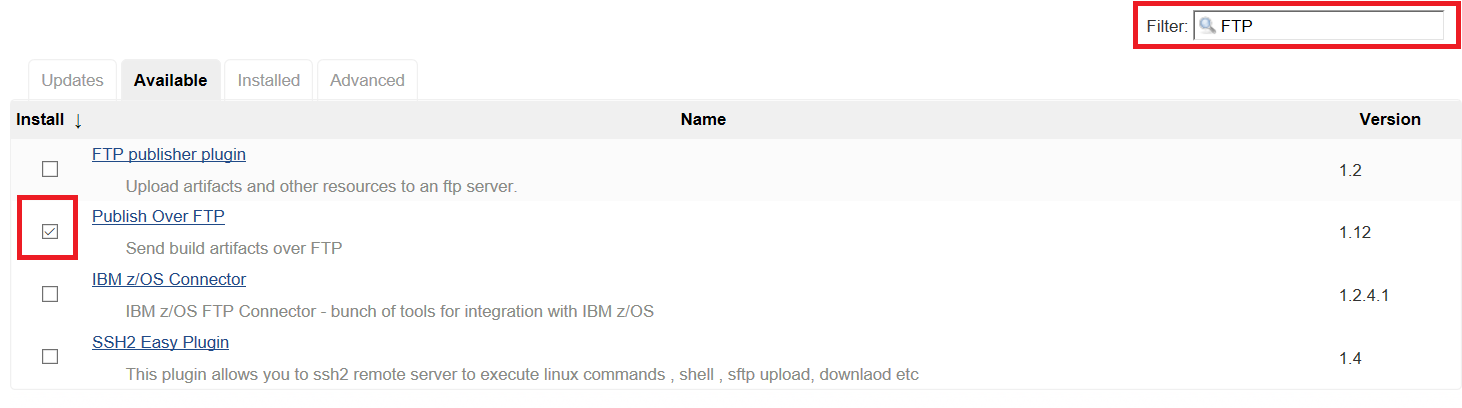
1. Dentro de sua instância Jenkins, vá em “*Manage Jenkins*” e, dentro do menu de configuração, clique em “*Manage Plugins*”.



1. Clique na aba “Available”.



1. Use a caixa de pesquisa no canto superior direito e, após localizar os plugins indicados, marque a caixa de seleção ao lado dele, como por exemplo, o Plugin de FTP.

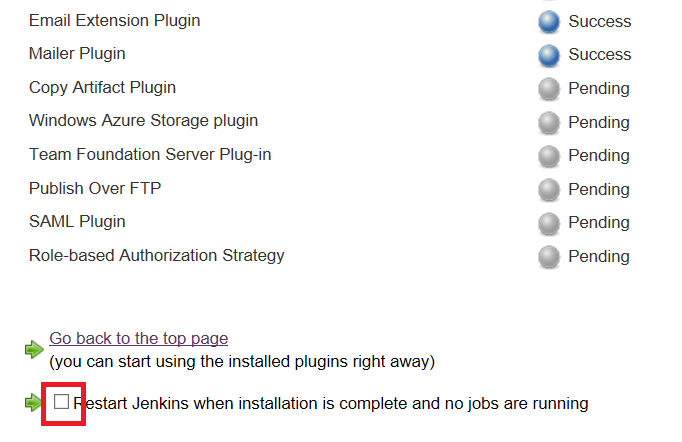


Selecione os seguintes plugins:

* [Publish Over FTP](https://plugins.jenkins.io/publish-over-ftp)
* [Windows Azure Storage plugin](https://plugins.jenkins.io/windows-azure-storage)
* [Team Foundation Server Plug-in](https://plugins.jenkins.io/tfs)
* [SAML Plugin](https://plugins.jenkins.io/saml)
* [Role-based Authorization Strategy](https://plugins.jenkins.io/role-strategy)

Após selecionar todos os plugins acima, clique no botão “*Install without restart*”, no fim da página.

1. Durante a instalação, marque a caixa indicando que quer reiniciar o Jenkins após a instalação dos plugins. O Jenkins será reiniciado.



1. Especificamente em nosso Lab, iremos compilar um site ASP.NET Core. Para isso, precisamos instalar os binários do SDK do .NET Core, versão 1.0.0-rc3-004530. Conecte-se na VM remota do Azure e faça a seguinte download dos binários:
   1. Windows:

Baixe o instalador <https://go.microsoft.com/fwlink/?linkid=839640>. O processo de instalação é bem simples, apenas avance os passos durante o instalador na VM Windows de destino via RDP.

* 1. Instalação no Ubuntu 16.04:

$ sudo sh -c 'echo "deb [arch=amd64] https://apt-mo.trafficmanager.net/repos/dotnet-release/ xenial main" > /etc/apt/sources.list.d/dotnetdev.list'

$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 417A0893

$ sudo apt-get update

$ sudo apt-get install dotnet-dev-1.0.0-rc3-004530

* 1. Para outros SOs:

baixe os binários do seguinte repositório: <https://github.com/dotnet/core/blob/master/release-notes/download-archives/rc3-download.md>. Lembre-se de colocar os binários em uma pasta referenciada pelo PATH de seu SO, ou atualize o mesmo para referenciar essa pasta.

1. Para checar que o binário foi instalado com sucesso, vá no bash ou cmd e digite o seguinte comando:

$ dotnet –version

Se a saída for igual a seguinte, tanto a instalação dos binários quanto o PATH foi atualizado com sucesso:

1.0.0-rc3-004530

1. Por fim, precisamos instalar o Git nas máquinas.
   1. Windows:

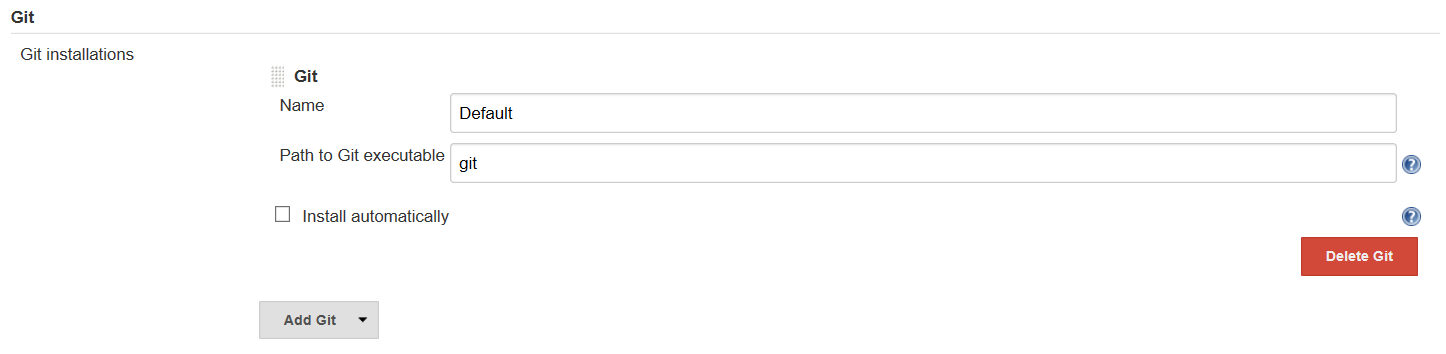
Para os usuários de Windows, necessariamente a instalação deverá ser feita, dado que o Git não vem por padrão nas máquinas. Seguindo o mesmo processo de instalação guiada que foi usada no .NET Core SDK. Baixe em <https://git-scm.com/download/win> e o instale depois.

* 1. Ubuntu 16.04:

Por padrão, a instalação Ubuntu 16.04 do Azure já vem com o Git instalado. Porém, caso esteja trabalhando em uma VM qualquer, dê o seguinte comando para instala-lo.

$ sudo apt-get install git

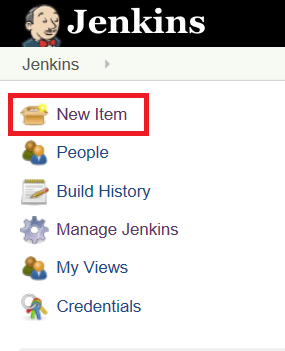
Para conferir se não há problemas com o PATH, vá em: Jenkins, Manage Jenkins e em seguida Global Tool Configuration. Cheque se não aparece nenhum erro, como na imagem:



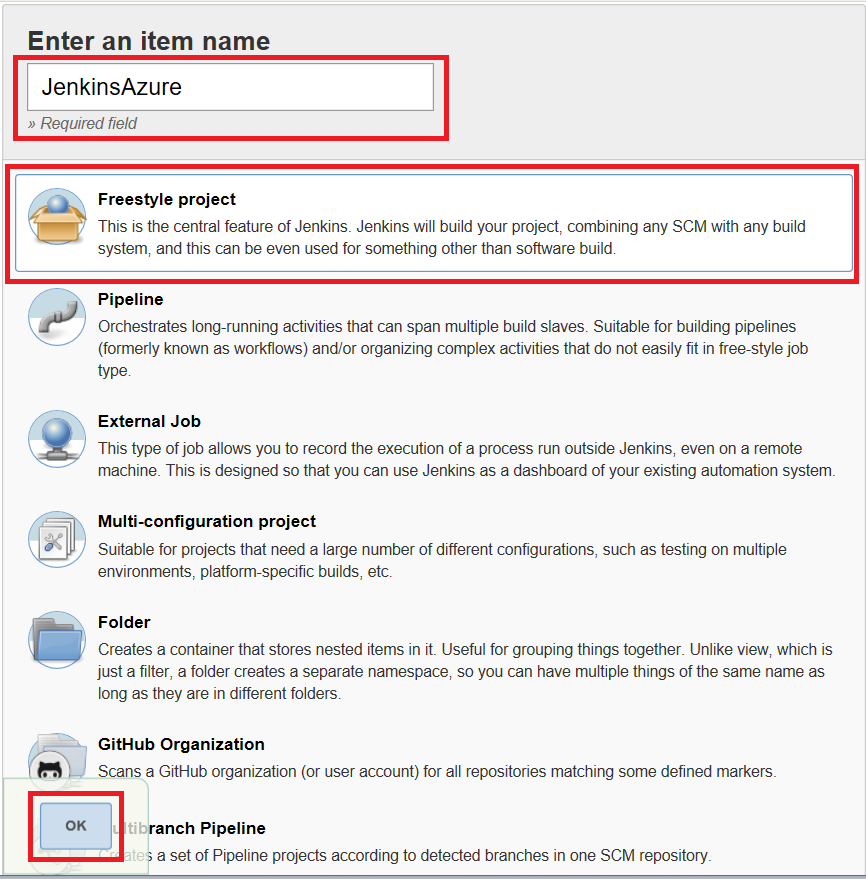
# Continuous Integration

Vamos fazer o setup de um simples job de build de um site em ASP.NET Core. O respectivo código-fonte está em <https://github.com/allantargino/JenkinsAzure>. Iremos configurar o Jenkins para, quando houver um commit em seu repositório, o processo de build será disparado.

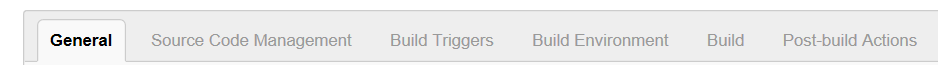
1. Logue-se em sua conta do Github e faça um fork do projeto <https://github.com/allantargino/JenkinsAzure>. Caso não saiba ou tenha dúvidas acerca do processo, visite este [guia](https://help.github.com/articles/fork-a-repo/). Ao fim desta etapa, você terá um endereço: https://github.com/SeuNomeDeUsuario/JenkinsAzure
2. Dentro do menu principal do Jenkins, clique em New Item.



1. Dê um nome para o Job que será criado e escolha o tipo de projeto como *Freestyle Project*. Depois clique em OK para cria-lo.

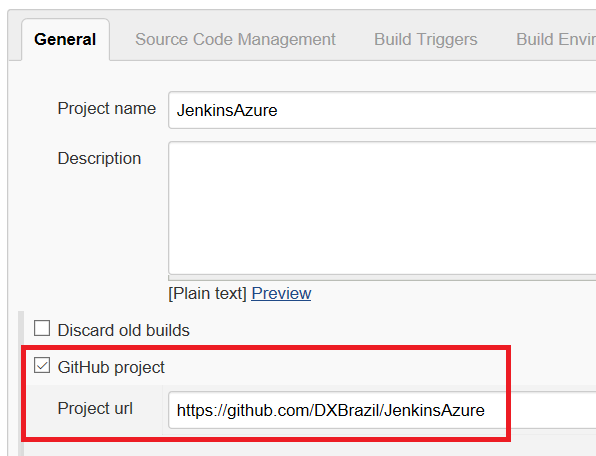


1. Temos as seguintes sessões dentro um Freestyle job:

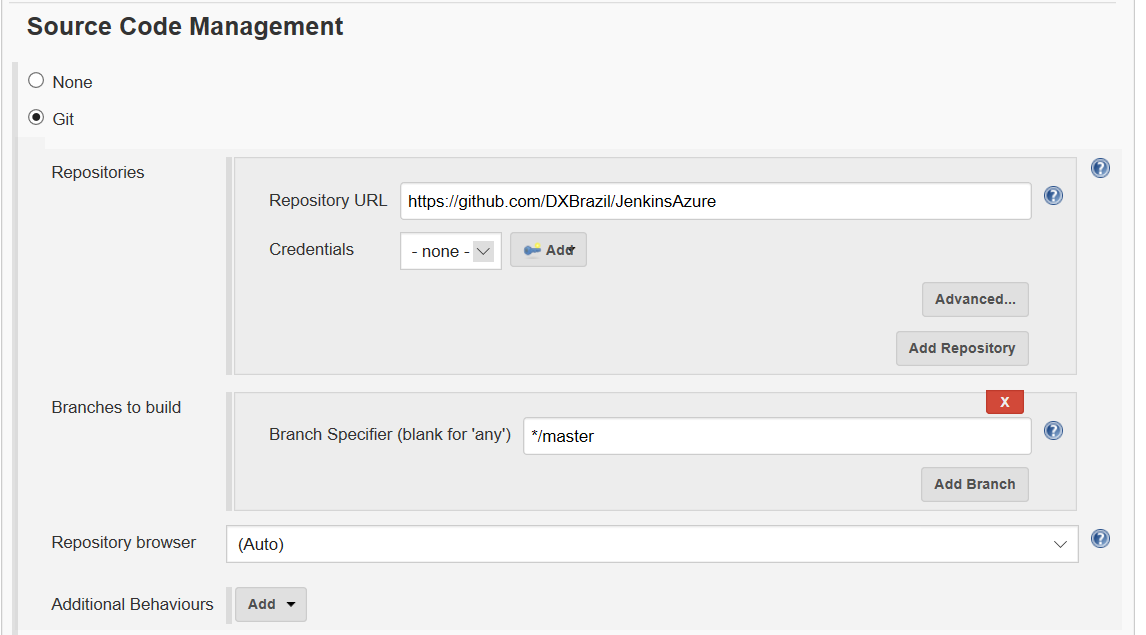


* + General: Informações gerais sobre o projeto.
  + Source Code Management: Iremos configurar onde o Jenkins deve procurar o código fonte a ser realizado o build.
  + Build Triggers: Parte onde iremos setar com qual frequência e como o Jenkins será disparado para fazer o processo de build.
  + Build Environment: Configurações envolvendo como o build afeta o ambiente local em que o build será feito.
  + Build: Passos e comandos de build (ou outros genéricos) a serem orquestrados pelo Jenkins.
  + Post-build Actions: Após ter os artefatos de build, pode-se decidir o que fazer com eles, como guarda-los localmente ou publicar uma aplicação, por exemplo.

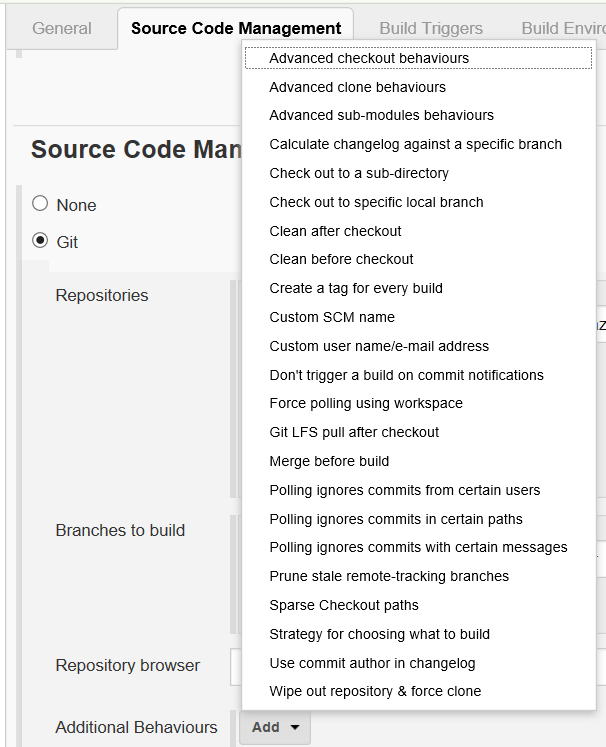
1. (opcional) Dentro da aba General, marque a opção GitHub Project e cole o link do seu projeto para ter o acesso fácil de dentro da página de resumo de seu build .



1. Em Source Code Management, escolha a opção Git e, no campo Repository URL, cole o link do **seu fork** do repositório no GitHub. Mantenha a branch master como sendo o alvo do build.

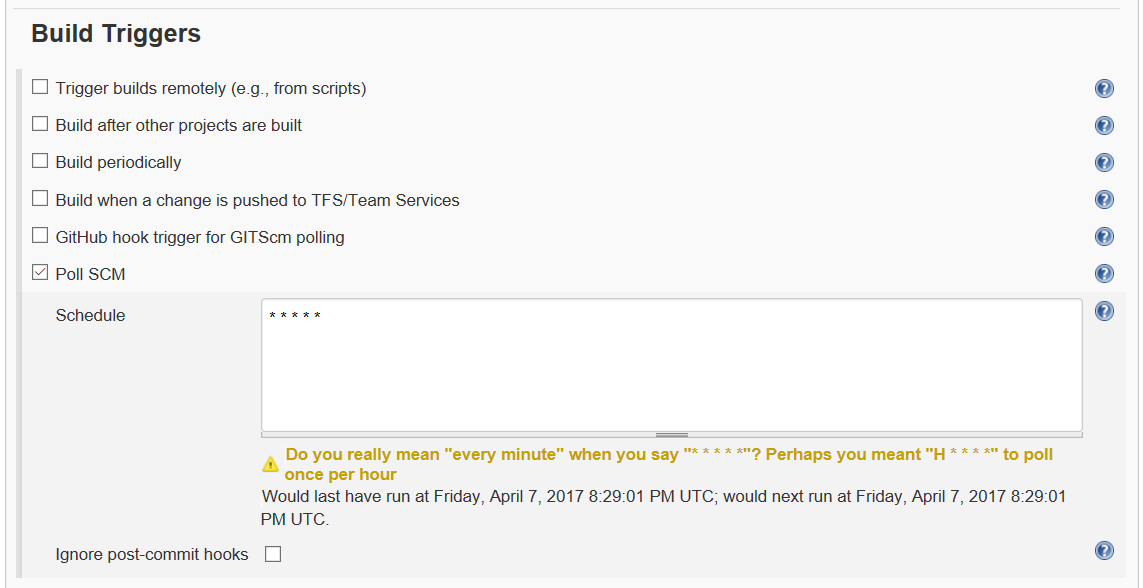


Por padrão iremos deixar todas as outras opções inalteradas. Mas pode-se ver que o plugin Git é um dos mais poderosos da plataforma do ponto de vista de CI, dado os vários behaviours/condições que podem ser adicionadas. Vale muito a pena explorar algumas dessas opções depois.



1. Há algumas maneiras de se configurar qual a frequência e quais os triggers que irão disparar o processo de build a ser configurado adiante. Pode-se esperar um build ser concluído de um outro projeto, pode-se fazer o build periodicamente, fazer o pooling em um repositório Git/VSTS, entre outros. Vamos fazer a opção de pooling em nosso SCM (Source Control/Code Management), fazendo o processo de CI de fato

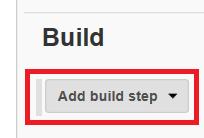
Escolha a opção Poll SCM e como argumento do Schedule, digite, sem as aspas: “\* \* \* \* \*”. Isso significa que o polling será feito a cada minuto.



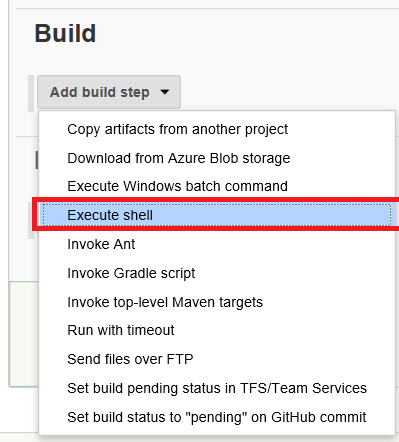
Essa opção de schedule basicamente irá executar o comando de pooling segundo uma frequência definida na sintaxe Cron. Para mais detalhes sobre Cron e alguns exemplos, clique [aqui](https://www.pantz.org/software/cron/croninfo.html). Mas de uma maneira bem rápida, cada um dos dígitos corresponde, respectivamente, aos campos Minute, Hour, Day of Month, Month e Day of Week. Por exemplo:

* + - 1. \* \* \* \* \* → A cada minuto
      2. 0 0 1 \* \* → A cada mês (dia 01, as 00h00 AM)
      3. 0 0 \* \* \* → A cada dia (as 00h00 AM)

1. Na sessão de Build, clique em Add build step.

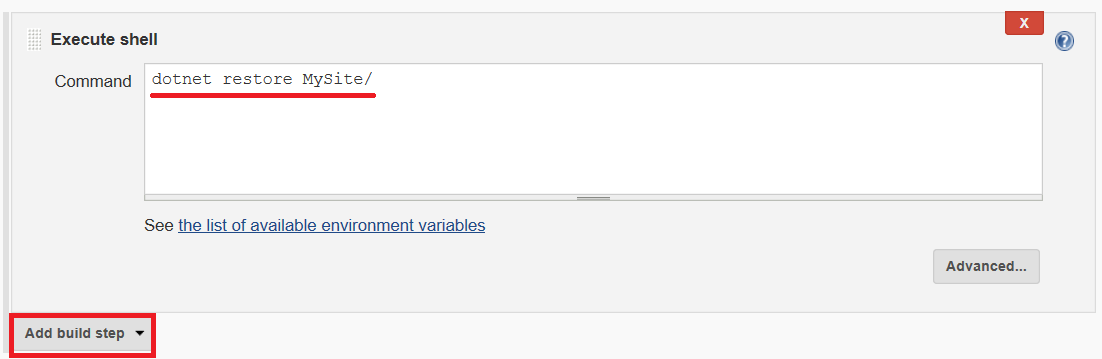


Um menu contendo todas ações de build irá aparecer. Uma característica do Jenkins é que, conforme novos plugins vão sendo adicionados, a UI vai mudando junto. Este menu contem e mostra alguns dos plugins que instalamos. Mas lembre-se, na maioria das vezes que não houve um plugin que faça o trabalho por você, uma linha de comando quase sempre poderá fazer! Clique em, se estiver no **Linux**, *Execute Shell*, ou caso esteja no **Windows**, *Execute Windows batch command*.



Cole o seguinte comando dentro deste primeiro passo de build:

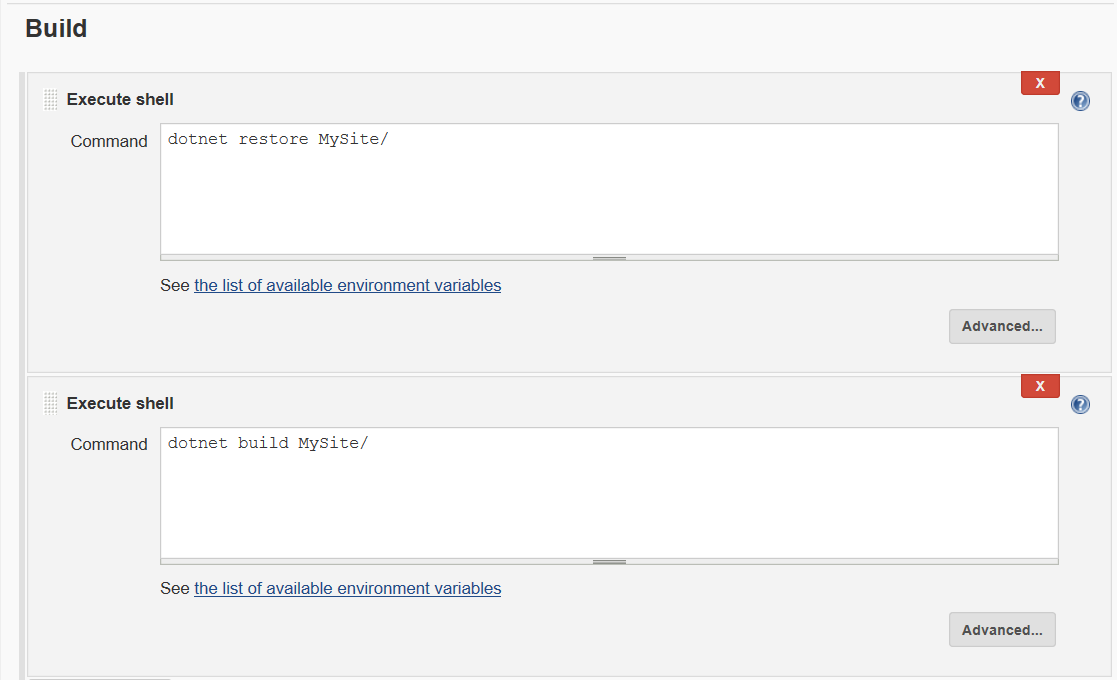
dotnet restore MySite/

Ele será responsável por olhar as dependências dentro do arquivo de configuração de nosso projeto e restaura-las. Em seguida, adicione novamente um passo de, se estiver no Linux, *Execute Shell*, ou caso esteja no Windows, *Execute Windows batch command*:

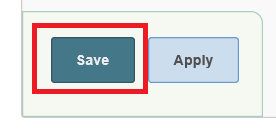
Desta vez, adicione o comando:

dotnet build MySite/

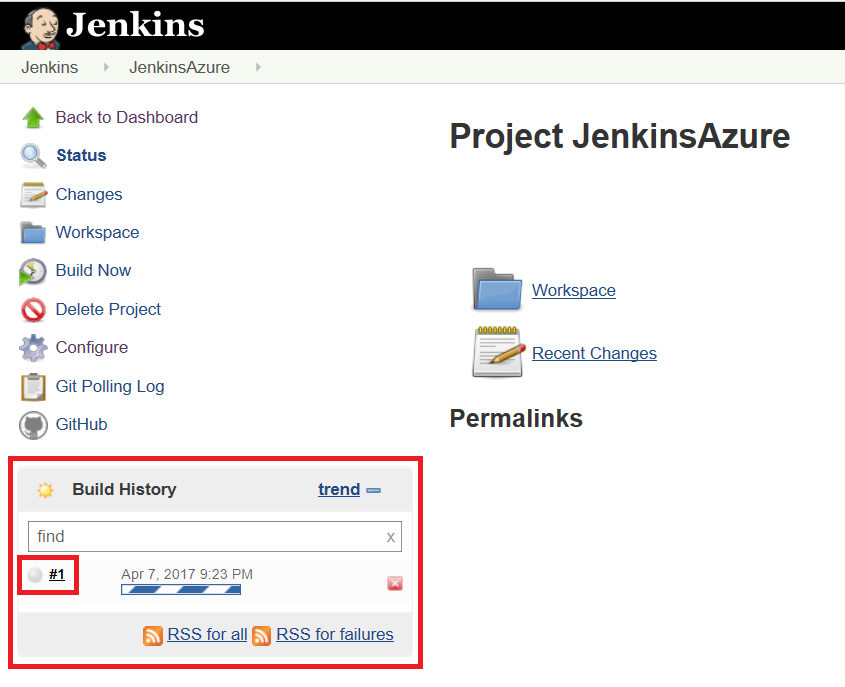
Ele irá realizar a compilação propriamente dita do nosso projeto.



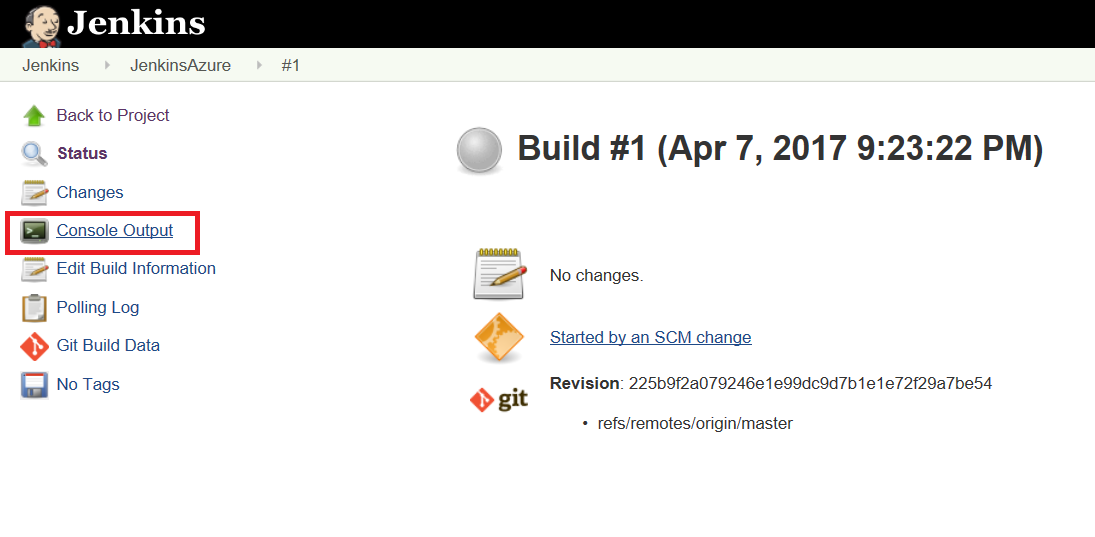
Clique em salvar:



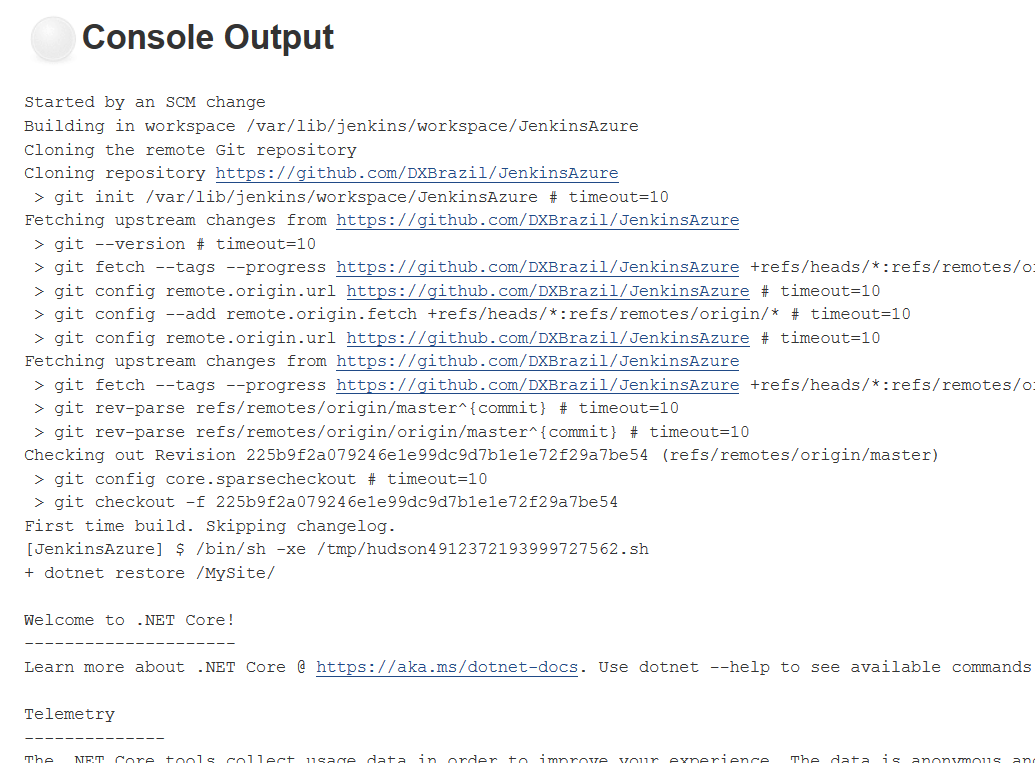
1. Uma vez que nosso CI esteja configurado com sucesso, após clicar em salvar, o primeiro build de nosso projeto será enfileirado após alguns segundos.



Ao clicar no link #1, você será levado a página de informações acerca deste build. Clique em Console Output para ver em tempo real o que está acontecendo com seu processo de build.



A primeira vez que o comando dotnet restore roda em uma máquina, ele precisa realizar alguns processos internos que demoram um pouco. As próximas builds serão extremamente mais rápidas.

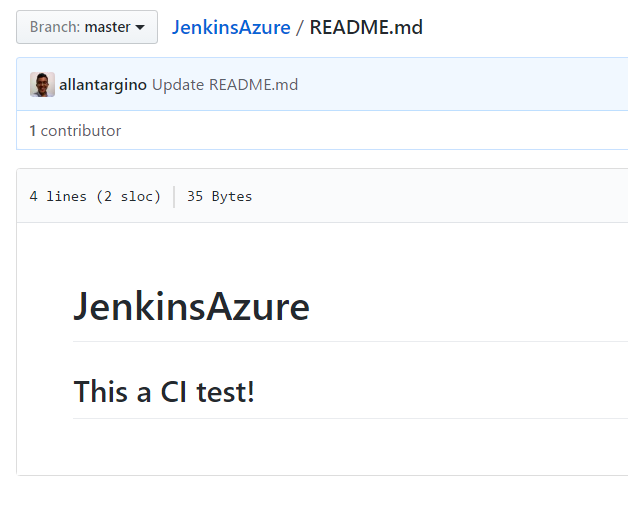


Uma vez que o processo de build tenha finalizado, você pode checar se o mesmo foi bem sucedido ou não, de acordo com a cor da bolinha de status associado a ele, onde azul indica sucesso e vermelho indica falha.



1. O que testamos até agora não diz respeito ao processo de CI ainda, dado que o Jenkins deve responder após algum commit em nosso repositório. Para forçar isso vamos editar o arquivo README.md para disparar um commit simples e forçar um build.

Vá em seu fork do repositório e adicione qualquer linha ao arquivo README.md e faça o commit do mesmo.



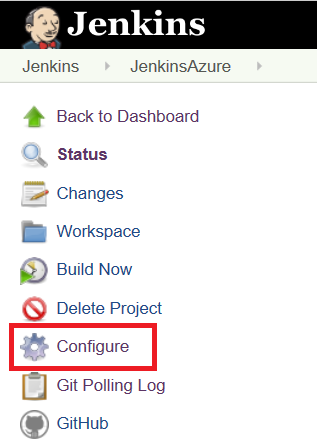
Ao visitar a instância Jenkins, você notará dentro da página do projeto que um segundo build (#02) foi disparado (e muito provavelmente) já concluído (de tão rápido e pequeno que é o build).



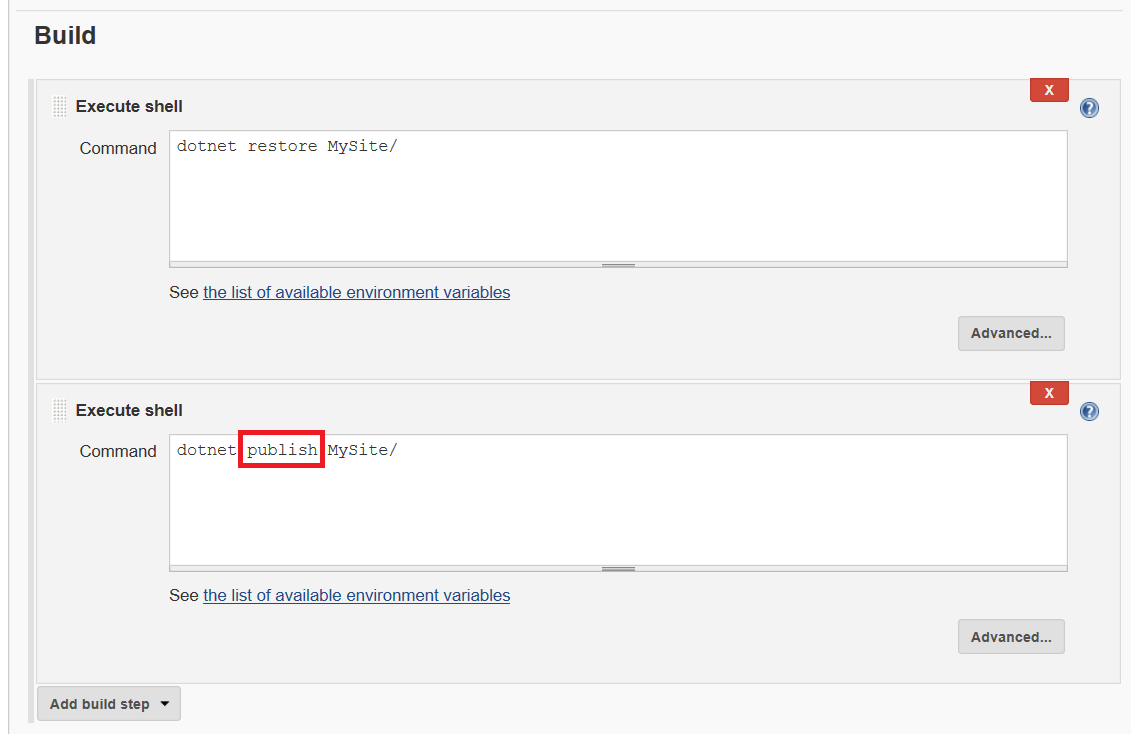
# Continuous Delivery

Vamos adicionar passos de publicação de artefatos em nosso processo de *build*, garantindo assim que, para cada *commit* disparado no sistema, um *build* seja disparado e o respectivo resultado do site seja publicado direto em produção (neste caso especifico).

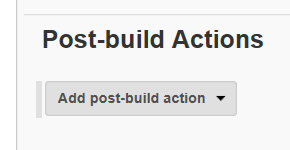
1. Dentro da página do projeto, clique em *Configure* para editarmos as definições do build.



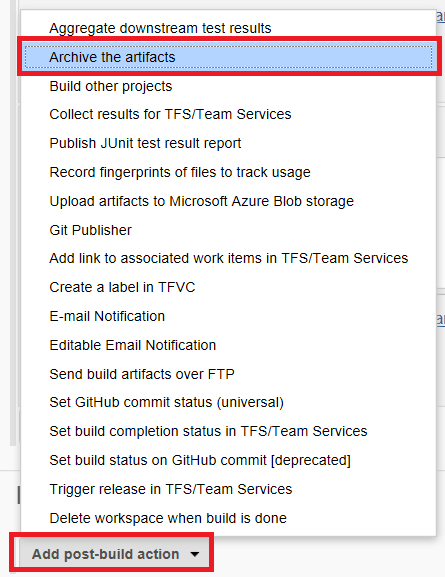
1. Vá até a sessão de Build, e troque, dentro do segundo passo, a palavra *build* por *publish*.



1. Dentro de Post-build Actions, teremos um comportamento similar ao de Build, com ações nativa e outras influenciadas diretamente pelos plugins instalados em sua instância.



Clique em Add post-build action, seguido de Archive the artifacts.

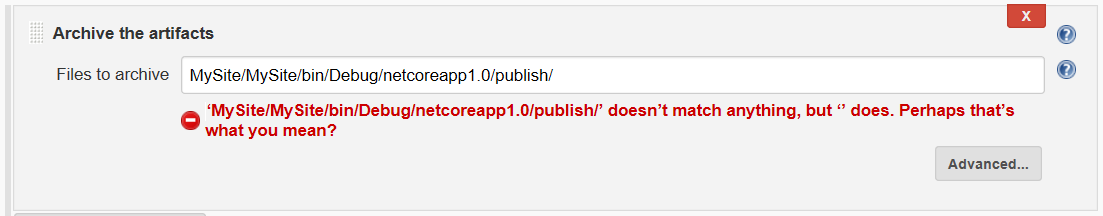


Adicione o seguinte caminho

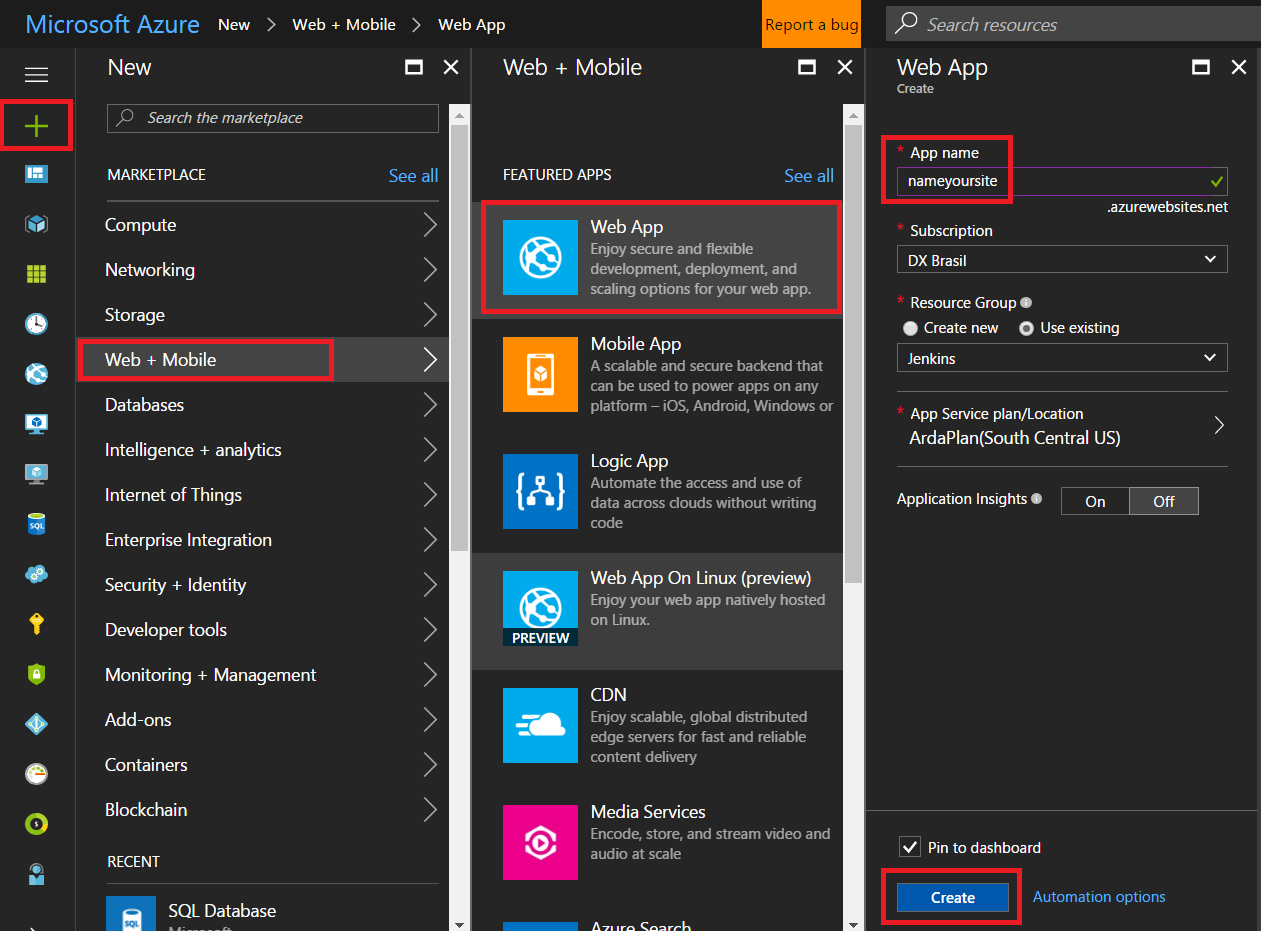
MySite/MySite/bin/Debug/netcoreapp1.0/publish/

Nesta pasta serão guardados os arquivos que deveremos publicar em nosso site na nuvem. Ele mostrará um erro indicando que não encontrou a pasta, mas é por que ainda não rodamos o processo de publicação nenhuma vez ainda.

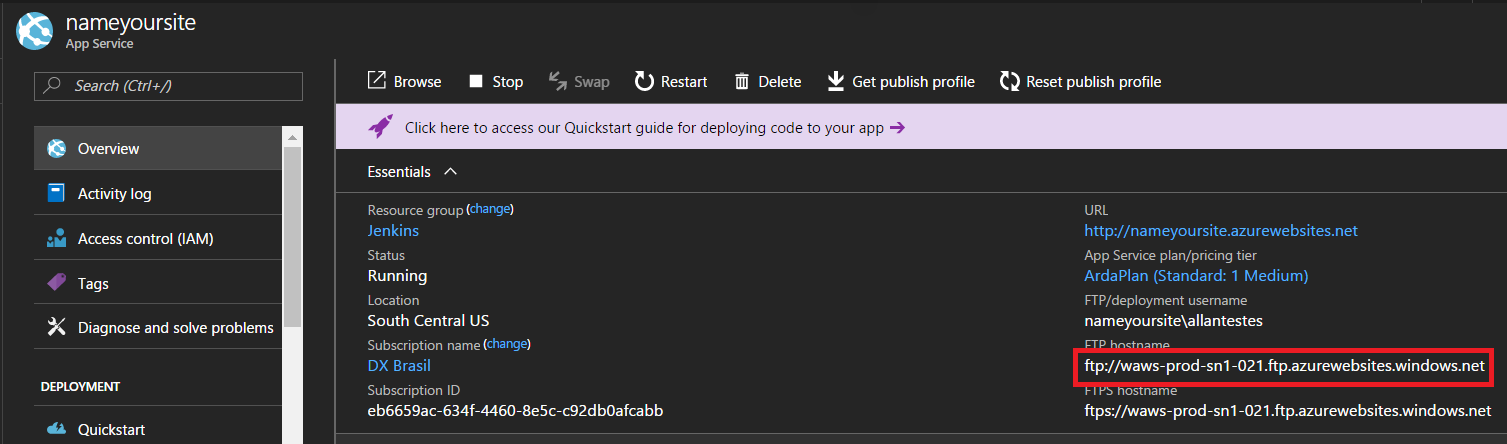
Uma vez que o processo de build tenha rodado, teremos os artefatos em nossa máquina local. Mostraremos mais à frente como armazenar os artefatos na nuvem com alta disponibilidade.

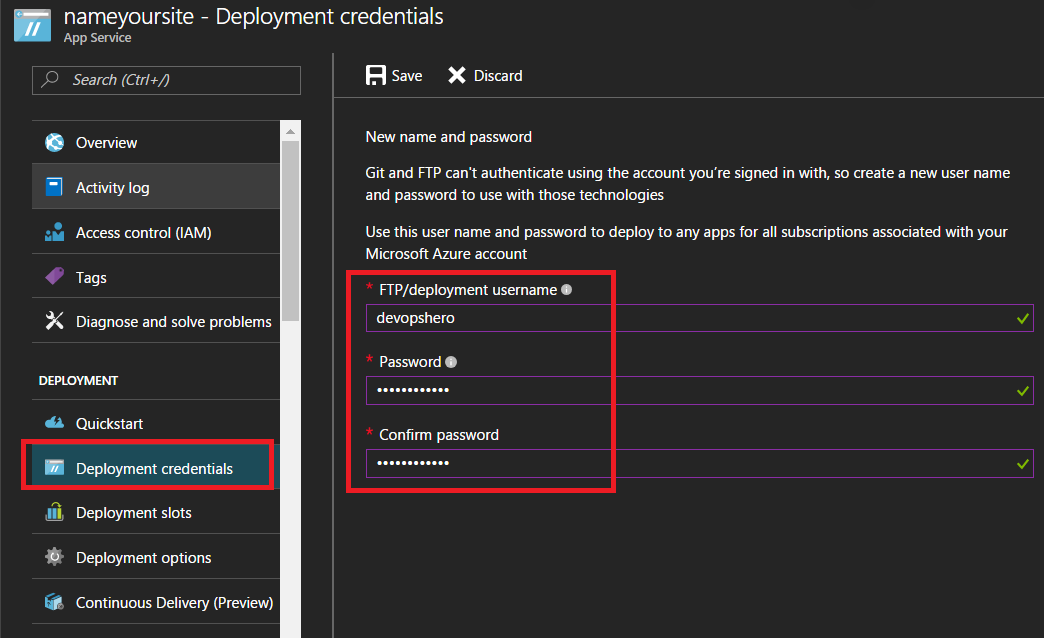


1. Vamos voltar ao portal do Azure para criar um Web App para hostear nosso site. Vá em <https://portal.azure.com>. Veja a imagem abaixo para auxilia-lo na criação de um Web App.



Após a criação do Web App, guarde o endereço do host de FTP.





# Guardando artefatos com alta disponibilidade no Azure Blob Storage

# Adicionando a segurança do Azure AD ao nosso servidor

# Integração e orquestração de Builds através do VSTS

# Usando do poder de extensibilidade do Jenkins e do Azure

## Usando a REST API do Jenkins para projetos ou integrações personalizadas

[Trecho em desenvolvimento: em breve estará terminado!]

## Usando as REST APIs do Azure com etapas de Build/Release do Jenkins

Definitivamente um ponto forte do Jenkins é a sua variabilidade do acervo de plugins. Porém, com a velocidade que um movido a APIs funciona dificilmente você terá plugins que atenda todos os pré-requisitos de um projeto. Novamente, a saída é usar scripts de linha de comando para fazer o trabalho sujo. Em especial, temos um poder gigantesco de automação ao considerar usar o Jenkins para automatizar infraestrutura no Azure usando suas APIs.

Para isso, instale o Azure CLI 2.0 em: <https://docs.microsoft.com/en-us/cli/azure/install-azure-cli>

Vá em seu terminal e digite o seguinte comando para criar uma conexão com o Azure. Siga as instruções para fazer a conexão com sucesso.

az login

Após isso, só adicionar scripts de automação/build em sua instância Jenkins. Alguns exemplos de casos de uso incluem:

* + - [Create a virtual machine](https://docs.microsoft.com/en-us/azure/virtual-machines/scripts/virtual-machines-linux-cli-sample-create-vm-quick-create?toc=%2fcli%2fazure%2ftoc.json)
    - [Create a VM with Docker enabled](https://docs.microsoft.com/en-us/azure/virtual-machines/scripts/virtual-machines-linux-cli-sample-create-docker-host?toc=%2fcli%2fazure%2ftoc.json)
    - [Create a VM and run configuration script](https://docs.microsoft.com/en-us/azure/virtual-machines/scripts/virtual-machines-windows-cli-sample-create-vm-iis?toc=%2fcli%2fazure%2ftoc.json)
    - [Create a web app and deploy code to a staging environment](https://docs.microsoft.com/en-us/azure/app-service-web/scripts/app-service-cli-deploy-staging-environment?toc=%2fcli%2fazure%2ftoc.json)
    - [Scale a web app manually](https://docs.microsoft.com/en-us/azure/app-service-web/scripts/app-service-cli-scale-manual?toc=%2fcli%2fazure%2ftoc.json)
    - [Create a single database and configure a firewall rule](https://docs.microsoft.com/en-us/azure/sql-database/scripts/sql-database-create-and-configure-database-cli?toc=%2fcli%2fazure%2ftoc.json)
    - [Scale a single database](https://docs.microsoft.com/en-us/azure/sql-database/scripts/sql-database-monitor-and-scale-database-cli?toc=%2fcli%2fazure%2ftoc.json)

Para mais exemplos, [clique aqui](https://docs.microsoft.com/en-us/azure/virtual-machines/linux/cli-samples?toc=%2fcli%2fazure%2ftoc.json&bc=%2fcli%2fazure%2fbreadcrumb%2ftoc.json).

# Referências

<Pegar todos links usados aqui>

Jenkins: The Definitive Guide”, de John Ferguson Smart, O'Reilly Media – 2011

(Pegar link dele)

<https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Ubuntu>