Amadis

WHITE PAPER

Deploying application to Linux VM's using Jenkins and Azure DevOps Services

Continuous Integration (CI) and Continuous Deployment (CD) form a pipeline by which you can build, release, and deploy your code. Azure DevOps Services provides a complete, fully featured set of CI/CD automation tools for deployment to Azure. Jenkins is a popular third-party CI/CD server-based tool that also provides CI/CD automation. You can use Azure DevOps Services and Jenkins together to customize how you deliver your cloud app or service.

In this white paper we are going to demonstrate on how to use Jenkins to build a web app and Azure DevOps services to deploy it.

Creating a Jenkins Master:

Before we start the process we need access to a Jenkins Server, so we have to first configure Jenkins master on an Azure virtual machine. In the following example we are going to see how to install Jenkins on an Ubuntu Linux VM with the tools and plug-ins configured to work with Azure.

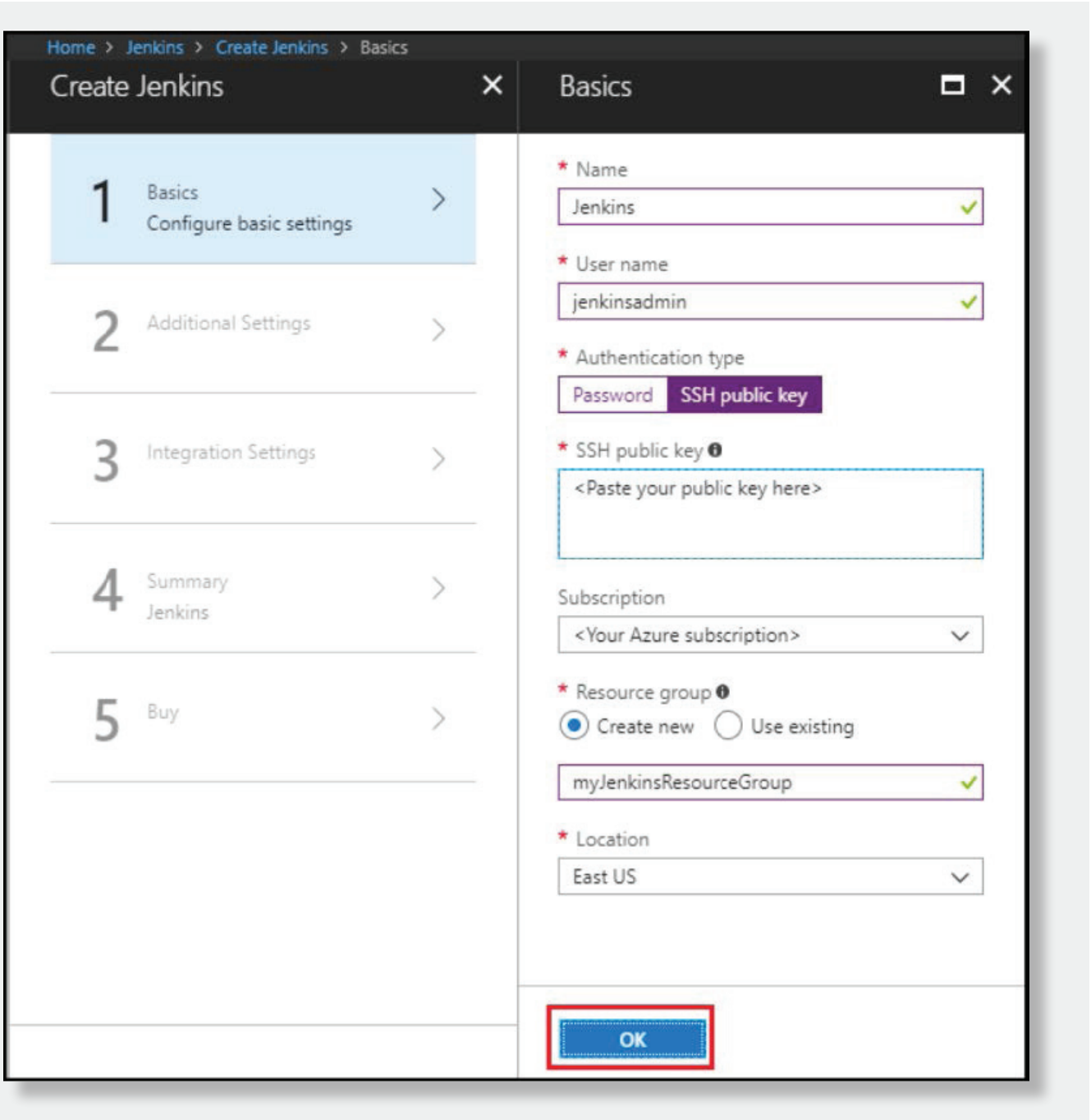
Jenkins supports a model where the Jenkins server delegates work to one or more agents to allow a single Jenkins installation to host a large number of projects or to provide different environments needed for builds or tests. The following steps will guide you through installing and configuring a Jenkins server on Azure.

- In your browser, open the Azure Marketplace image for Jenkins.
- Select **GET IT NOW**.
- After reviewing the pricing details and terms information, select **Continue.**
- Select **Create** to configure the Jenkins server in the Azure portal.

In the Basics tab, specify the following values:

- **Name** - Enter Jenkins.
- **User name** - Enter the user name to use when signing in to the virtual machine on which Jenkins is running. The user name must meet specific requirements.
- **Authentication type** - Select **SSH public key**.
- **SSH public key** - Copy and paste an RSA public key in single-line format (starting with ssh-rsa) or multi-line PEM format. You can generate SSH keys using ssh-keygen on Linux and macOS, or PuTTYGen on Windows. For more information about SSH keys and Azure, see the article, How to Use SSH keys with Windows on Azure.

- **Subscription** - Select the Azure subscription into which you want to install Jenkins.
- **Resource group** - Select **Create new**, and enter a name for the resource group that serves as a logical container for the collection of resources that make up your Jenkins installation.
- **Location** - Select **East US**.
- **Select OK to proceed to Additional settings tab.**



In the **Additional Settings** tab, specify the following values:

- **Size** - Select the appropriate sizing option for your Jenkins virtual machine.
- **VM disk type** - Specify either HDD (hard-disk drive) or SSD (solid-state drive) to indicate which storage disk type is allowed for the Jenkins virtual machine.
- **Virtual network** - (Optional) Select **Virtual network** to modify the default settings.
- **Subnets** - Select S**ubnets**, verify the information, and select **OK**.
- **Public IP address** - The IP address name defaults to the Jenkins name you specified in the previous page

with a suffix of -IP. You can select the option to change that default.
- **Domain name label** - Specify the value for the fully qualified URL to the Jenkins virtual machine.
- **Jenkins release type** - Select the desired release type from the options: LTS, Weekly build, or Azure Verified. The LTS and Weekly build options are explained in the article, Jenkins LTS Release Line. The Azure Verified option refers to a Jenkins LTS version that has been verified to run on Azure.
- **JDK Type** - JDK to be installed. Default is Zulu tested, certified builds of OpenJDK.

Select OK to proceed to integrations settings tab.

In the **Integration Settings** tab, specify the following values:

- **Service Principal** - The service principal is added into Jenkins as a credential for authentication with Azure. Auto means that the principal will be created by MSI (Managed Service Identity). Manualmeans that the principal should be created by you.
  - **Application ID** and **Secret** - If you select the Manual option for the **Service Principal** option, you'll need to specify the Application ID and Secret for your service principal. When creating a service principal, note that the default role is **Contributor**, which is sufficient for working with Azure resources.
- **Enable Cloud Agents** - Specify the default cloud template for agents where ACI refers to Azure Container Instance, and VM refers to virtual machines. You can also specify No if you don't wish to enable a cloud agent.

Select **OK** to proceed to the **Summary** tab.

- When the **Summary** tab displays, the information entered is validated. Once you see the **Validation passed** message (at the top of the tab), select **OK**.
- When the **Create** tab displays, select **Create** to create the Jenkins virtual machine. When your server is ready, a notification displays in the Azure portal.

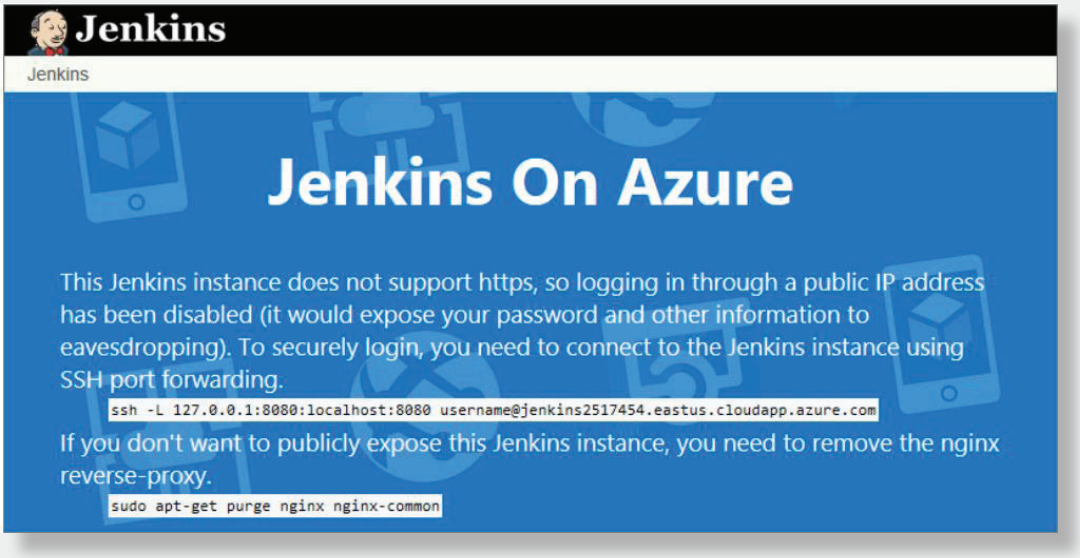
Connecting to Jenkins:

Navigate to your virtual machine (example, http://jenkins2517454.eastus.cloudapp.azure.com/) in your web browser. The Jenkins console is inaccessible through unsecured HTTP so instructions are provided on the page to access the Jenkins console securely from your computer using an SSH tunnel.

Set up the tunnel using the ssh command on the page from the command line, replacing username with the name of the virtual machine admin user chosen earlier when setting up the virtual machine from the solution template.

After you have started the tunnel, navigate to http://localhost:8080/ on your local machine.

Get the initial password by running the following command in the command line while connected through SSH to the Jenkins VM.



Unlock the Jenkins dashboard for the first time using this initial password.

Select **Install suggested plugins** on the next page and then create a Jenkins admin user used to access the Jenkins dashboard.

Configuring Jenkin Plugins:

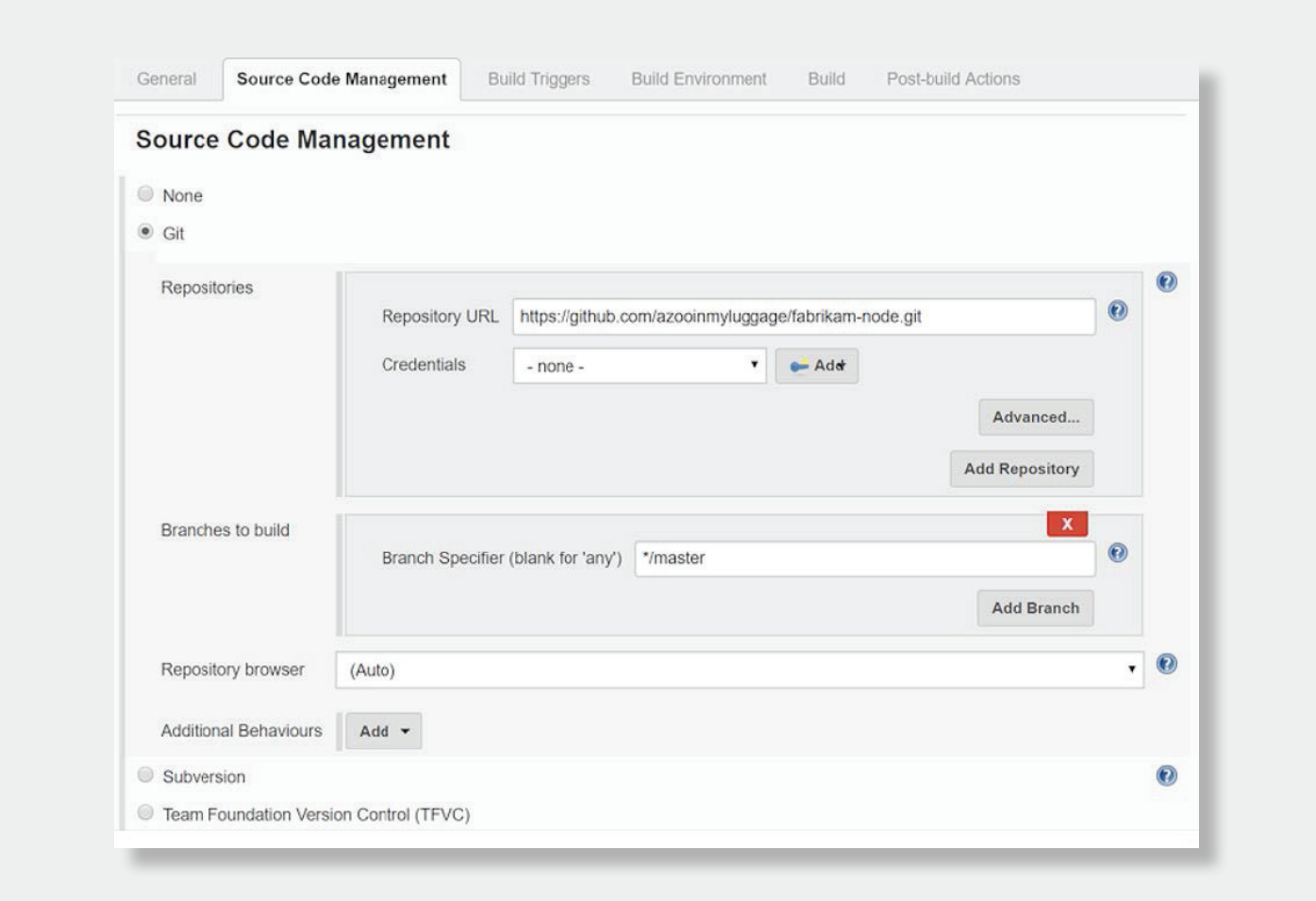
Now that we have configured Jenkins master on a Linux server and connected to Jenkins console we have to install the Jenkin plugins needed to make Continuous Integration and Continuous Deployment work. For this example, we would be using a Node.js web application working in Express framework. This is sample script is available in GitHub.

The two plugins that we are going to install in this example are **NodeJS** and **VS Team Services Continuous Deployment**. Please follow the below steps for the same.

- Open your Jenkins account and select **Manage Jenkins.**
- On the **Manage Jenkins** page, select **Manage Plugins**.
- Filter the list to locate the **NodeJS** plug-in, and select the **Install without restart** option.
- Filter the list to find the **VS Team Services Continuous Deployment** plug-in and select the **Install without restart** option.
- Go back to the Jenkins dashboard and select **Manage Jenkins**.
- Select **Global Tool Configuration**. Find **NodeJS** and select **NodeJS installations**.
- Select the **Install automatically** option, and then enter a **Name** value.
- Select **Save.**

Configuring Jenkins Freestyle project for Node.js:

1. Select **New Item**. Enter an item name.
2. Select **Freestyle project.** Select **OK**.
3. On the **Source Code Management** tab, select **Git** and enter the details of the repository and the branch that contain your app code.



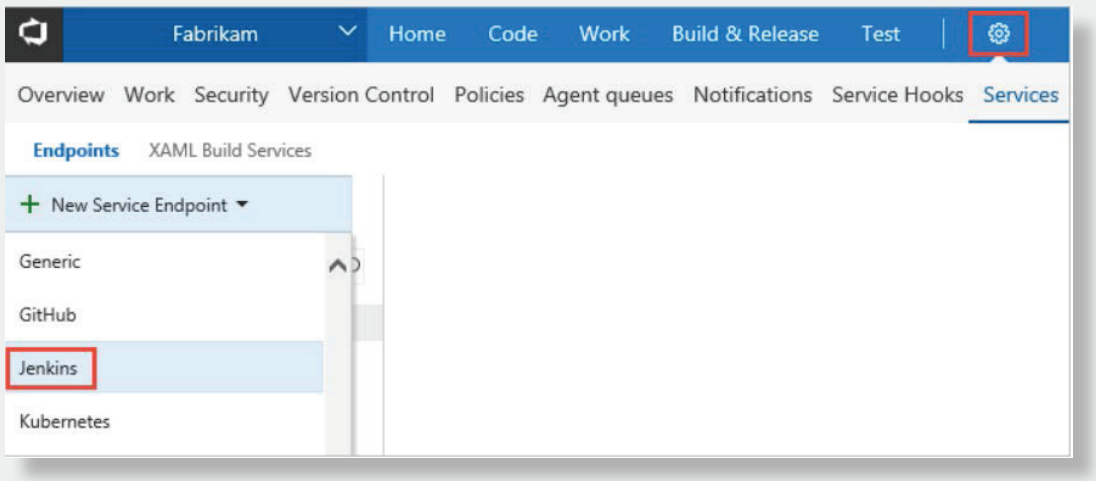
4. On the **Build Triggers** tab, select **Poll SCM** and enter the schedule H/03 * * * * to poll the Git repository for changes every three minutes.
5. On the **Build Environment** tab, select **Provide Node & npm bin/ folder PATH** and select the **NodeJS Installation** value. Leave **npmrc file** set to **use system default.**
6. On the **Build** tab, select **Execute shell** and enter the command npm install to ensure that all dependencies are updated.

Configuring Jenkins for Azure DevOps Services integration:

1. Create a PAT in your Azure DevOps Services organization if you don't already have one. Jenkins requires this information to access your Azure DevOps Services organization. Be sure to store the token information for upcoming steps in this section.
2. In the **Post-build Actions** tab, select **Add post-build** action. Select **Archive the artifacts**.
3. For **Files to archive**, enter **/* to include all files.
4. To create another action, select **Add post-build action**.
5. Select **Trigger release in TFS/Team Services.** Enter the URI for your Azure DevOps Services organization, such as **https://{your-organization-name}.visualstudio.com**.
6. Enter the **Project** name.
7. Choose a name for the release pipeline. (You create this release pipeline later in Azure DevOps Services.)
8. Choose credentials to connect to your Azure DevOps Services or Team Foundation Server environment:
   - Leave **Username** blank if you are using Azure DevOps Services.
   - Enter a username and password if you are using an on-premises version of Team Foundation Server.

Creating a Jenkins service endpoint:

- Open the **Services** page in Azure DevOps Services, open the **New Service Endpoint** list, and select **Jenkins**.
- Enter a name for the connection.
- Enter the URL of your Jenkins server, and select the **Accept untrusted SSL certificates** option. An example URL is **http://{YourJenkinsURL}.westcentralus.cloudapp.azure.com**.
- Enter the username and password for your Jenkins account.
- Select **Verify connection** to check that the information is correct.
- Select **OK** to create the service endpoint.



Creating a Jenkins service endpoint:

You need a deployment group to register the Azure DevOps Services agent so the release pipeline can be deployed to your virtual machine. Deployment groups make it easy to define logical groups of target machines for deployment, and to install the required agent on each machine.

1. Open the **Releases** tab of the **Build & Release hub**, open **Deployment groups**, and select **+ New**.
2. Enter a name for the deployment group, and an optional description. Then select **Create**.
3. Choose the operating system for your deployment target virtual machine. For example, select **Ubuntu 16.04+**.
4. Select **Use a personal access token in the script for authentication**.
5. Select the **System prerequisites** link. Install the prerequisites for your operating system.
6. Select **Copy script to clipboard** to copy the script.
7. Log in to your deployment target virtual machine and run the script. Don't run the script with sudo privileges.
8. After the installation, you are prompted for deployment group tags. Accept the defaults.
9. In Azure DevOps Services, check for your newly registered virtual machine in **Targets** under **Deployment Groups**.

Creating release pipeline in Azure Devops:

A release pipeline specifies the process that Azure Pipelines uses to deploy the app. In this example, you execute a shell script.

To create the release pipeline in Azure Pipelines:

1. Open the **Releases** tab of the **Build & Release** hub, and select **Create release pipeline**.
2. Select the **Empty** template by choosing to start with an **Empty process**.
3. In the **Artifacts** section, select **+ Add Artifact** and choose **Jenkins** for **Source type**. Select your Jenkins service endpoint connection. Then select the Jenkins source job and select **Add**.
4. Select the ellipsis next to **Environment 1**. Select **Add deployment group phase.**
5. Choose your deployment group.
6. Select **+** to add a task to **Deployment group phase**.
7. Select the **Shell Script** task and select **Add**. The **Shell Script** task provides the configuration for a script to run on each server in order to install Node.js and start the app.
8. For **Script Path**, enter **$(System.DefaultWorkingDirectory)/Fabrikam-Node/deployscript.sh**.
9. Select **Advanced**, and then enable **Specify Working Directory**.
10. For **Working Directory**, enter $**(System.DefaultWorkingDirectory)/Fabrikam-Node**.
11. Edit the name of the release pipeline to the name that you specified on the **Post-build Actions** tab of the build in Jenkins. Jenkins requires this name to be able to trigger a new release when the source artifacts are updated.
12. Select **Save** and select **OK** to save the release pipeline.

Execute manual and CI-triggered deployments

1. Select **+ Release** and select **Create Release**.
2. Select the build that you completed in the highlighted drop-down list, and select **Queue**.
3. Choose the release link in the pop-up message. For example: “Release **Release-1** has been created.”
4. Open the **Logs** tab to watch the release console output.
5. In your browser, open the URL of one of the servers that you added to your deployment group. For example, enter **http://{your-server-ip-address}**.
6. Go to the source Git repository and modify the contents of the **h1** heading in the file app/views/index.jade with some changed text.
7. Commit your change.
8. After a few minutes, you will see a new release created on the **Releases** page of Azure DevOps. Open the release to see the deployment taking place.

You have now successfully used Jenkins and Azure Devops services for CI/CD process.

Amadis
Guhan Palanisamy
Cloud Solutions Engineer, Amadis Technologies
Phone: 862-300-9516
Email: guhanp@amadisglobal.com

100 Overlook Center, 2nd Floor, Princeton, NJ 08540
Phone: +1 609 375 2755
Email: info@amadisglobal.com