**Chapter 4**

Scaling Tutum Containers

< OTHER CHAPTERS ARE ON THE WAY ☺ >

**Arun Das**

Graduate Research Assistant

**Open Cloud Institute**

The University of Texas

San Antonio, Texas

# SCALE EASILY WITH TUTUM

Tutum makes working with containers easy. Here is what we are going to learn in this section:

1: What scaling means

2: Methods to scale

3. Scaling using Tutum GUI / Web Interface

3: Scaling using Tutum CLI

4: Scaling using YAML files.

## Introduction

Tutum is a Container Platform, which facilitates a simpler way of working with containers. It helps to build, deploy and manage apps across any cloud. Tutum provides a GUI, which is accessible from the internet, to carry out all the steps we otherwise need to give in though CLI. This gives much more clarity of the current state of your cloud, along with it a simpler straightforward way of building and deploying your app from ground up.

### Scaling

When you successfully launch your service without changing any of the default parameters, what you created is a service running on a single container. In order to create a redundant environment, which is much more resilient to involuntary issues causing unresponsive containers, we should scale our containers. As the name suggests, scaling of containers will make multiple copies of the same parent container. This ensures that even if one container fails, there is other backups to get your application running continuously.

### Methods to Scale

Scaling can be done through both the GUI web interface, through API and through the Tutum CLI. In this tutorial, we will cover scaling in GUI and CLI, with specific examples. One thing to keep in mind is that the CLI is much more powerful and versatile than the GUI. For multiple deployments and complex configurations, it is better to have a one line code or a script rather than doing multiple button clicks in your mouse or trackpad. Effective utilization of both GUI and CLI is what matters and this creates the perfect environment for Tutum environment.

### Scaling using Tutum GUI / Web Interface

In the previous chapters we leant how to create Repositories, bring your own Chameleon Baremetal / KVM Node into Tutum, start Services and to launch an app using your own repo. For the tutorials, we used default parameters wherever possible. Now, to enjoy the benefits of scaling, we need to increase and specify the number of containers that our Services have up and running. For this, let’s do the following steps:

|  |
| --- |
| NOTE: The tutorial is written believing that you have already completed Tutorials 1, 2 and 3. You should have a running one container Service in your Tutum by now. |

#### 1. Log in to your Tutum Web account by navigating to <http://www.tutum.co>

2. Click on the “Services” Tab

You should see your Service in the Service Dashboard. The Name of the service, its status and the image used can be seen here. If you do not have a service listed, you did not do the previous tutorials correctly. Please follow them from the beginning to proceed.

|  |
| --- |
| NOTE: If your service is “Stopped”, go ahead and run it. |

3. Click on your service’s name to get detailed info about it. You will be redirected to a similar webpage.

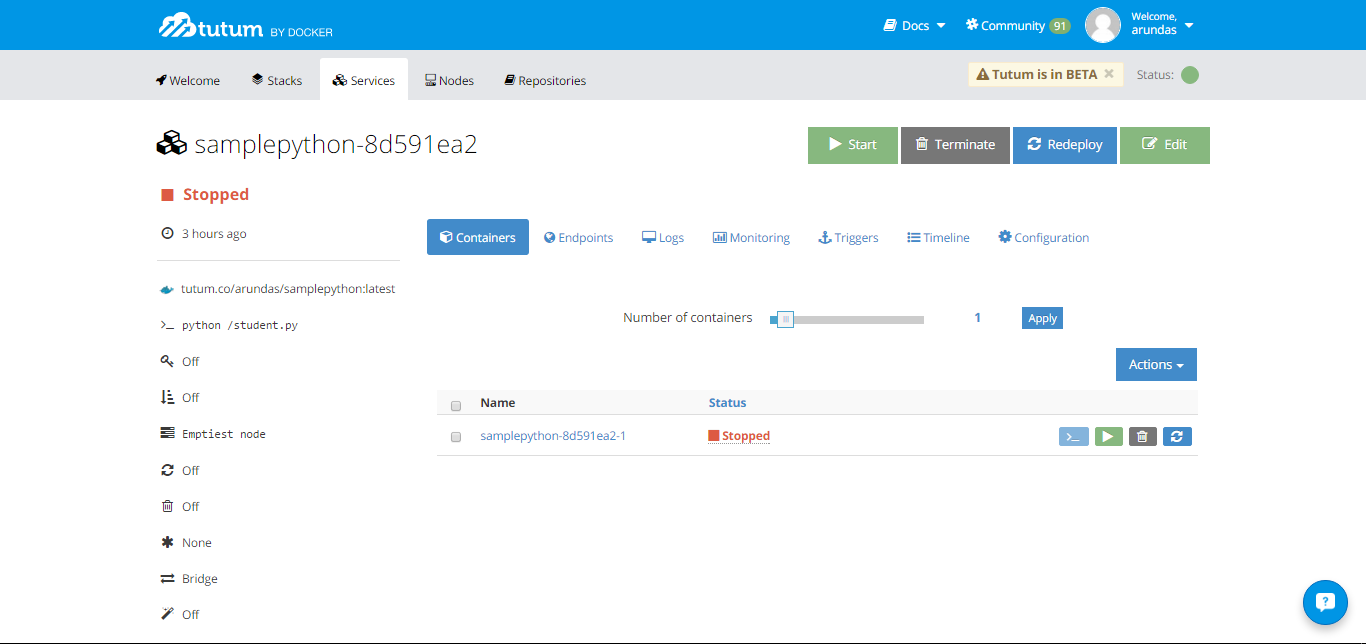


Fig. 1. Screenshot of one Service in Web GUI

The set parameters of your service is listed at the left hand side. At the top right is buttons to Start, Terminate or Redeploy your service. The Edit button lets you edit the parameters of your service.

4. The slider “Number of containers” lets you specify the number of containers for the particular service. This is what we use to scale our containers. Move the slider to the right and set the number of containers to 5 and click on “Apply”.

You can see that 4 more containers is getting started, making the total number of containers running the service to be 5. Tutum is this simple, for simple apps like in our example.

5. Now we will reduce the containers to 3. Go ahead and move the slider to the left and make the number of containers to 3. Hit “Apply”.

Voila! You just terminated 2 running containers successfully without affecting the proper working of others. Play around with different values of containers, make different services and try out the same.

### Scaling using Tutum CLI

SSH into your Chameleon instance running Tutum using your private key. Before going into and scaling our existing service, let us take a look at creating a new service with multiple containers. The following allows us to create one:



Here, while creating the new service, -t specifies the number of containers in the new service. The specifier --name is for the new services name. The image address is specified one space after the name scaled-container-service.

In order to scale the already existing service, we will use the following code format:



The identifier can be the service’s UUID (either long or short) or name[.stack\_name]. The parameter target-num-containers specify the number of containers to scale this service to. In our case, if we intend to scale our newly created service named scaled-container-service to 9 containers, use the following code:



Here the identifier is scaled-container-service and the target-num-containers is 9. This ends our discussion on scaling using Tutum CLI. Note that we can also scale down the number of containers by specifying it in the above code.

### Scaling using Scripts / YAML files

When you are working with multiple services and multiple containers, a simple script is more useful. The concept of Stack arises here. Stack is a collection of services that together make an application. A stackfile is a YAML based script file, which can create multiple services, each having one or many containers. We use a similar YAML based script in out tutorial to scale the containers.

The following is the script file used. The file is saved as tutum.yml in the home directory of the Chameleon node running Tutum.



deploymultiple is the name of the service created. The image used is specified with an image tag. target\_num\_containers specify the number of containers to be created for the specific service. The service is set to restart always. The emptiest node is used to deploy the service. All containers are set to deploy in parallel and the Service will be redeployed automatically.



The code creates a stack, in our case with only one service named deploymultiple and the specifier –f tells to read the file tutum.yml.

For more information on Tutum CLI and API refer the [Tutum API Documentation](https://docs.tutum.co/v2/api/?shell#introduction).