**What is Cloudwave - A cloud migration tool**

Bluewave project is a solution for migrating your infrastructure. Migration can be from DC to DC, DC to cloud, Cloud to DC, Cloud to Cloud.

It's a swiss army knife for infrastructure migration. There are similar tools available in the market like Cloudendure and zerto.

Right now the functionalities are limited to below features

* Migration of infrastructure to Azure , AWS
* Agentless
* Environment discovery
* Automatic network creation and server deployment
* Linux server migration

**Tech stack**

The tool is build on below techstack

* Python
* Ansible
* Mongodb

Cloudwave can be accessed and used from a web portal. Services and payloads are written in python. Ansible is used to prepare the servers ready for migration. MongoDB collects informations of all the servers and status of the whole process.

**The process of building Cloudwave - A cloud migration tool**

**How it works?**

We have 5 steps in the complete process of migration of the server.

* Discover: It will discover the number of core, RAM size, hard disk size and partitions, network details
* Blueprint: Here we will select appropriate machine size and CIDR range for the network
* Clone: This step will clone the on-premise server to the S3 bucket
* Convert: We convert the cloned server image to an AMI
* Build: We build the network components and build the server with the converted AMI

**How to use Cloudwave migration tool?**

In this scenario, we are going to migrate a RedHat Linux server from one AWS account to another AWS account. You can follow the same procedure when you want to migrate your servers from on-prem to AWS cloud.

**Components**

* Replication server with enough privileges to deploy network components(VPC, Subnet, Routing tables, Security group), deploy EC2 machines, perform the ec2 import, S3 bucket read/write access. (Access can be granted using ec2 roles or AWS keys)
* MongoDB - It can be configured in the replication server itself

**Step 1 - Setup the replication server**

Log in to the AWS account where you want to migrate the servers and create a new EC2 instance with ubuntu 18.04 OS for the replication server. Install and configure MongoDB. Once you have configured run the below command to create a user in MongoDB.

$mongo

mongo> db.createUser({user: "migrationuser",pwd:"mygrationtool",roles:[{role:"readWrite",db:"migration"}]})

Above command will create a user named migrationuser with password mygrationtool. User migrationuser will have read-write access to the database migration. Now you have to install the dependencies mentioned in the requirements.txt file. You can do that by running below command

pip install -r requirements.txt

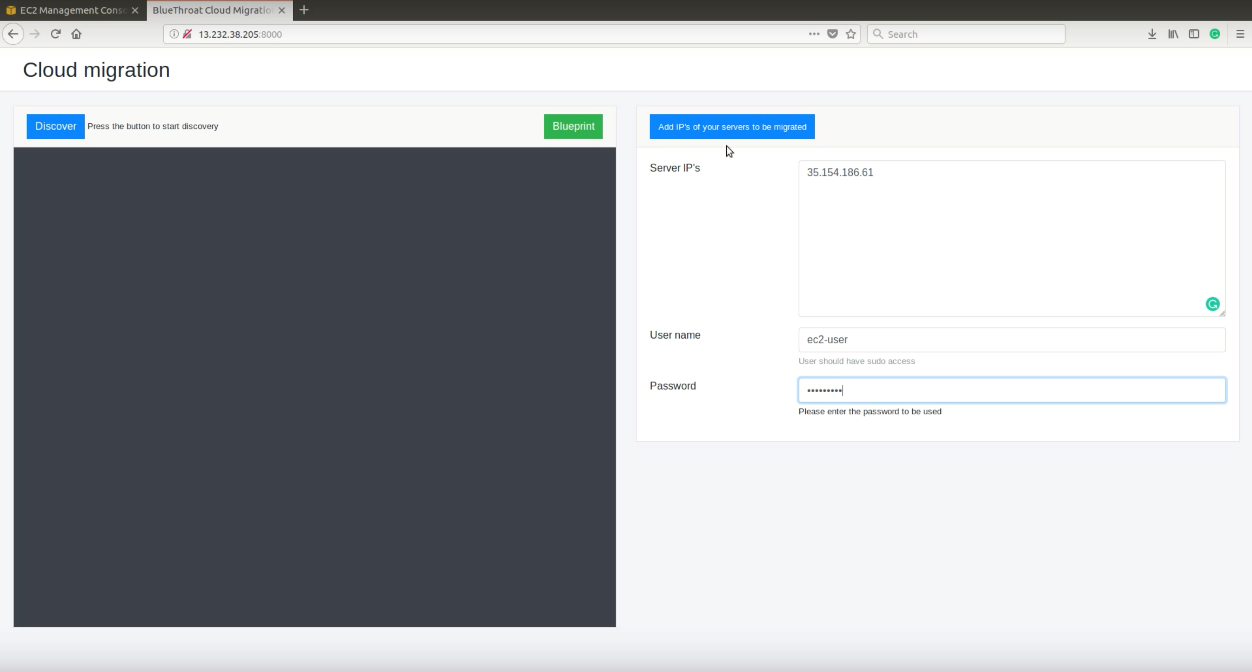
Once the dependencies are met you can start the application by running below command.

python app.py

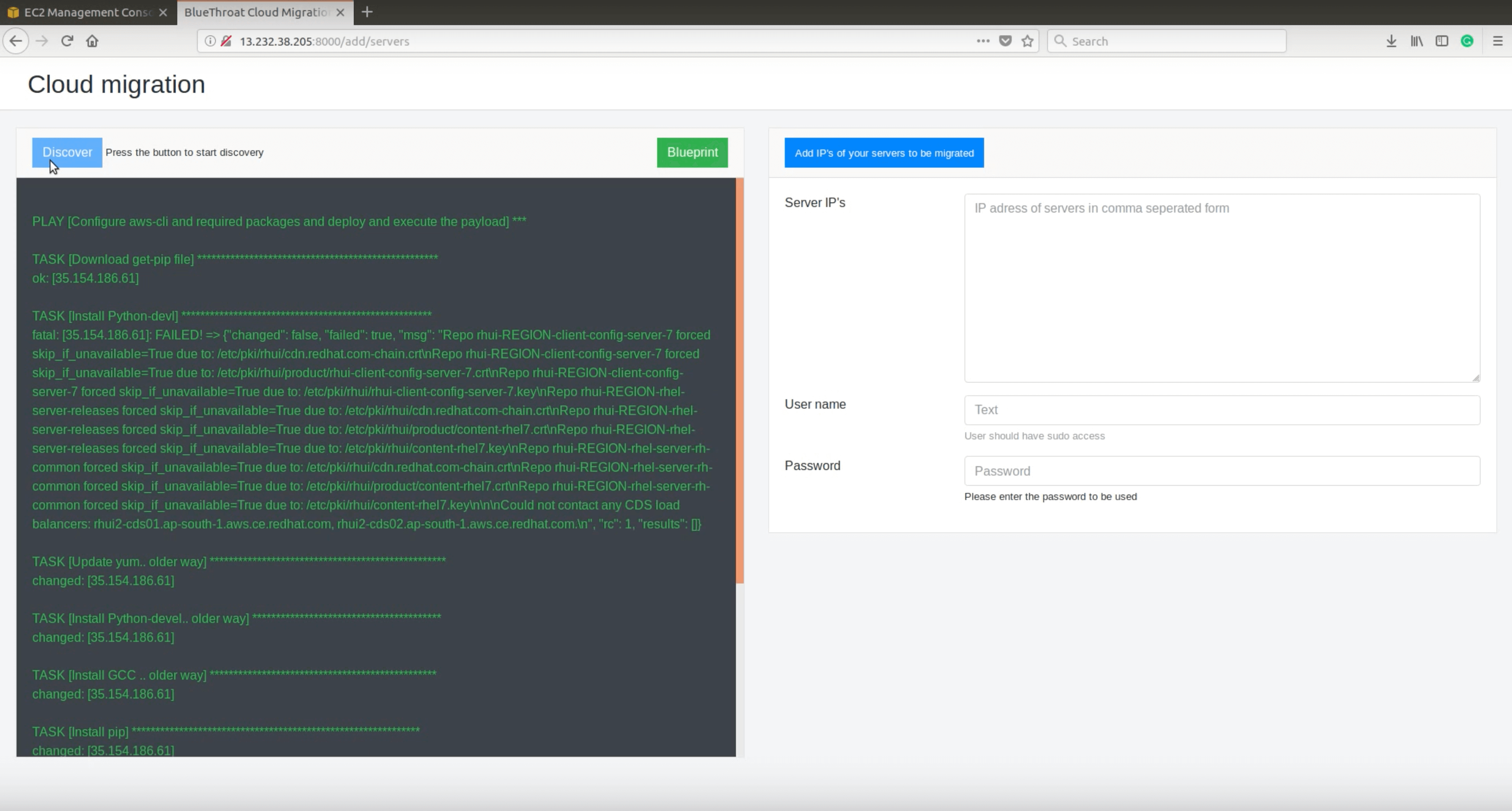
**Step 2 - Create access to the server which we want to migrate**

In this step, you have to create a user with sudo privileges on the server you want to migrate. We will be using ssh to connect with the server. Once it’s done we can move to the next step.

Step 3 - Start migrating your infrastructure

[](https://camo.githubusercontent.com/dfbbbce7faeb6fb5b26598742ad35a6e35c950e9/68747470733a2f2f63646e2e66696c65737461636b636f6e74656e742e636f6d2f4378764f326e79385255474b4c5171426a313964)

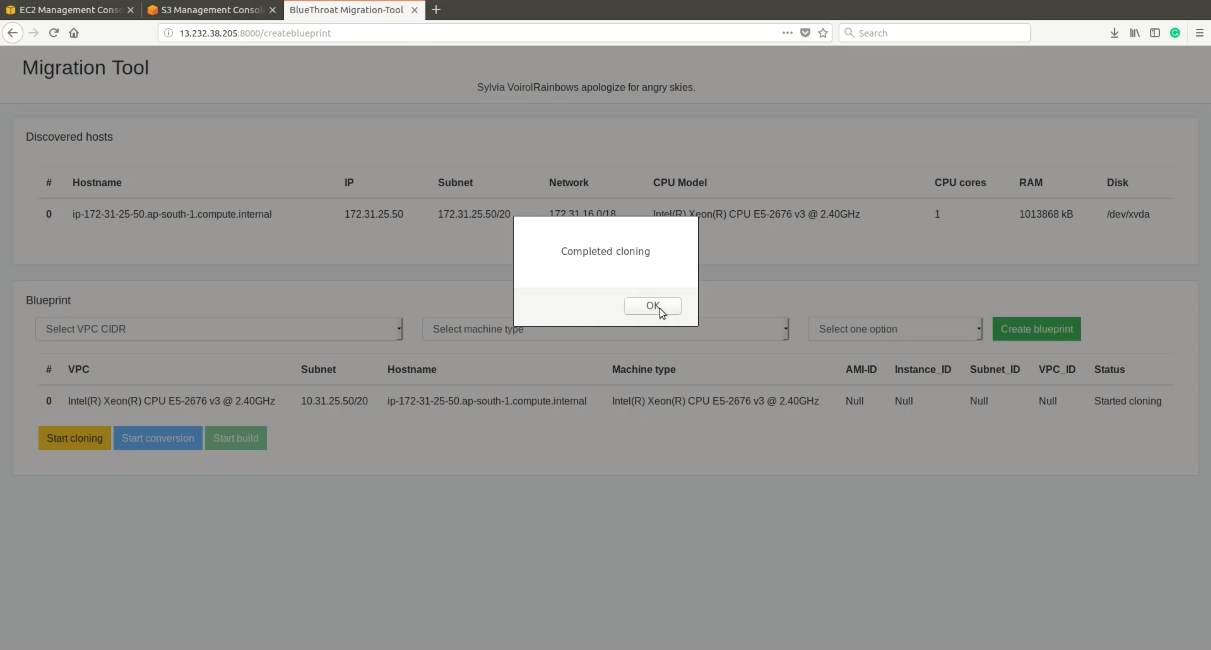
You have to add the servers you want to migrate with username and password to access those servers.

[](https://camo.githubusercontent.com/bce166e6a013334fa022fe9282d3ce32b3b330cb/68747470733a2f2f63646e2e66696c65737461636b636f6e74656e742e636f6d2f6178713030564e385276753143586a4659527074)

Once you have added the servers click on the button 'Add ips of your servers to be migrated' and then click on the 'Discover' button.

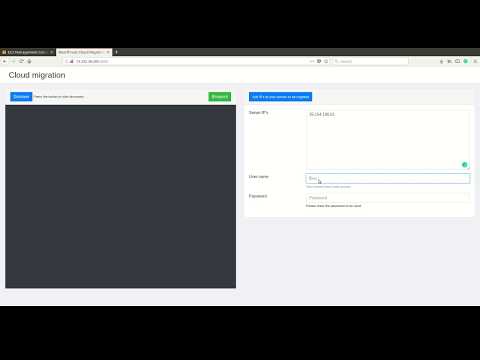
Once the discovery is completed it will show you an alert box with a message saying that discovery is completed. Then you have to click on the 'blueprint' button.

[](https://camo.githubusercontent.com/643071b48ebe1f5e5af81da415966f33da0ac3fc/68747470733a2f2f63646e2e66696c65737461636b636f6e74656e742e636f6d2f6c526a37716c7369534c434c6f63796c79793244)

In the blueprint page, you can see the details of the servers you added previously. As I already mentioned before, now the tool only supports DC to Azure ,AWS or AWS to AWS and Azure migration. So the pieces of information which you have to provide will be with respect to AWS. You have to select a VPC CIDR from the drop down, you have to select a machine type and you have to choose whether they are public facing or private machines. Once the dropdowns are selected, click on create blueprint button. Once you clicked the button it will create a network template to build. Then you have to press the start cloning button. [](https://camo.githubusercontent.com/6ced077d21085ac60ce3d9e7030227eab6b1498b/68747470733a2f2f63646e2e66696c65737461636b636f6e74656e742e636f6d2f61436157533949395439474935556a4b6e6d5267)

Once the cloning is completed you will get an alert box as above. At this step, your server's disk image will be copied in block level to the S3 bucket. Next, you have to click the Start conversion button. This will convert all the uploaded disk images to AMI in your Amazon account. Once you get the alert with the message Conversion completed, you have to click on Start build button, it will build VPC, subnet and deploy the machine to that subnet using the AMI's created in the previous step.

**Watch the video**

[](http://www.youtube.com/watch?v=xvpGDWLigW0)

**Challenges I faced**

When I started testing the tool, the only challenge I faced was getting an environment to test it out. All the testings were done in a local lab setup with my personal Azure , AWS account. I have tested applications servers with WordPress and MySql and the tool successfully migrated the server. And the application was working as before.

**Key learnings**

There are many things to be known before we do a data centre migration to the cloud. Network bandwidth, applications running, consumption of the resources and the number of servers are some of the key factors. Once the relevant information is available we can do the proper sizing of each server. We can decide when the business can afford downtime because we may need to stop the services running on database servers and according to the network bandwidth we can decide the number of servers needs to be migrated at a time. Basically, if we do all this manually it's really a tedious task. So I built the Cloudwave tool. Hope it will be useful to you too.

**Tips and advice**

Anything you do more than twice should be automated. Very basic devops rule :-). Automation makes your life easier. You will get more time to focus on multiple things.

**Final thoughts and next steps**

If you are really interested to contribute to this project below is the roadmap and you are most welcome. Together we can build a better tool for Cloud engineers.

**Roadmap**

* Zero touch sizing
* Migration of infrastructure to other vendors, GCP and Alibaba Cloud
* Automatic port detection and firewall creation
* Additional disk discovery and migration