

IERG4210 Tutorial 04

Deploying your webpage with elastic beanstalk

Shizhan Zhu

FYI

- If you still haven't got your AWS grant codes, please contact me asap!!!
(The homework task needs it!!)
- Topics to be covered in Phase 2:
 - About introduction to node.js and npm: Workshop on Tuesday (Feb. 3) will mainly focus on this;
 - About basic node project (and environment needed) set up, and deploying your files in the remote environment: main content for this tutorial.

Typical Steps for phase 2A.

- Step 1: basic installation -> components used in later steps
- Step 2: Run your hello world node project on your local machine (so it can be viewed in your local web browser)
- Step 3: Deploy the hello world node project on the remote instance (so it can be viewed around the world!)
- Step 4: (mainly covered in the work shop) Serve your shop page with the node project framework.

Content for today

- To deploy a “Hello world” page to the remote instance.
- Goal: we can view our released page via a public url! (No longer your local machine!)
- Pre-requisite: You should have finished Tutorial 3 task.
- This tutorial is a subset of finishing phase 2. After you get familiar with putting up your page (from this tutorial), you can go on to replace the Hello world project with your project (required in phase 2).
- Mostly using command, so less figures in this tutorial.
- Please frequently refer to <https://github.com/ierg4210/shop-samples>, great document.

Step 1: installation

- Python -> virtualenv
- Check if installed: `python --version` / `virtualenv --version`
- If yes: return the version codes, if no: command not found.
- Why do we need virtualenv by the way?
 - Avoid python configuration conflicts (dependence or versioning issues) between various projects. Projects in individual environment do not influence each other.
- Why do we need python by the way?
 - Because awsebcli is written in Python, and distributed via pip.
- If not installed:
 - For python download from <https://www.python.org/downloads/>
 - For virtualenv (after python installed): command: `sudo pip install virtualenv`

Step 1: Under virtual environment

- Git init your current dir (supposed to be where you develop the web page on your local machine.)
- Create a virtual env: `virtualenv "local-dev-env"` (Or other name you like)
- Enter the virtual env: `. local-dev-env/bin/activate`
- `.` in command line means to fetch all commands in the file onto the console and execute.
- You will see `(local-dev-env)` before your path now.
- Tell git not to record changes in local-dev-env folder (where python packages are installed). Put one line in file `.gitignore`: `local-dev-env/*`

Step 1: install the required packages

- Install awsebcli:
- Pip install awsebcli nodeenv
- Then install node.js using nodeenv:
- Nodeenv -p
- Use the -version (similar to python and virtualenv) to check if they are successfully installed.
- How to leave the virtual env? Command: deactivate
- Your project will be developed under the virtual env, and hence can only work under the virtual env.

Step 2: Hello world node project (local)

- Oh, what is npm?
- <https://docs.npmjs.com/getting-started/what-is-npm>, watch this interesting video first.
- During web development, projects are divided into several uncoupled packages, each performs a specific task like an expert. You can either fetch others' packages into your projects, or release yours to benefit others.
- Hence npm is the package manager.
- For node.js, npm is served as the default package manager.
- More details would be covered on Tuesday workshop.

Step2: the package.json

- It is the packaging format.
- <http://browsenpm.org/package.json>, click this interesting website, and you will get familiar with components within this file.
- You can make it in an interactive way:
- Command: `npm init`
- When prompt these questions, please set the value:
 - Is node project? Yes
 - Use git for version management? Yes
 - Entry point? `App.js`
 - For detail, you can set values according to <https://github.com/ierg4210/shop-samples/blob/master/SETUP-DEVJS.md>

Step 2: Implement the entry point: app.js

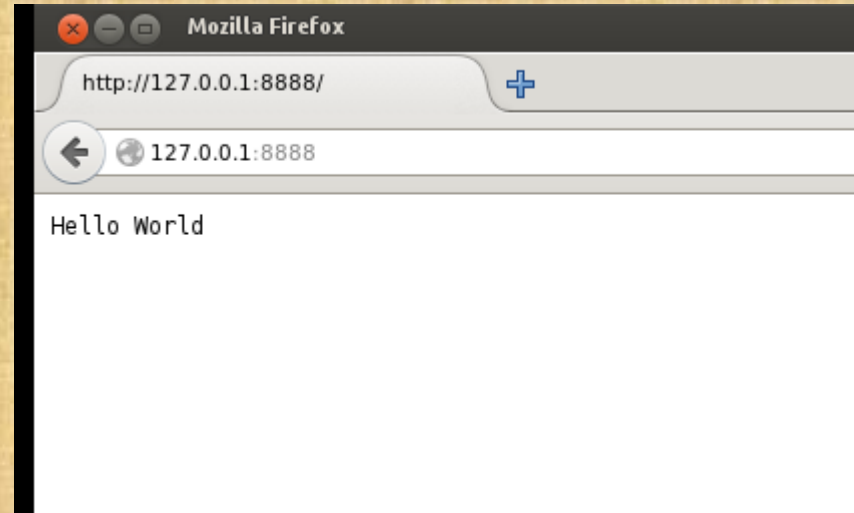
- For the hello world program, you can directly use the sample :

```
var http = require("http");  
http.createServer(function(request, response) {  
    response.writeHead(200, {"Content-Type": "text/plain"});  
    response.write("Hello World");  
    response.end();  
}).listen(process.env.PORT || 8888);
```

- Put them into your app.js file.
- Your local machine will listen to the given port, responded with an OK ¹⁰http.

Step 2: Check your local running status

- Command: `node app`
- You open a browser, type in `127.0.0.1:8888`, and you will find the hello word page there.

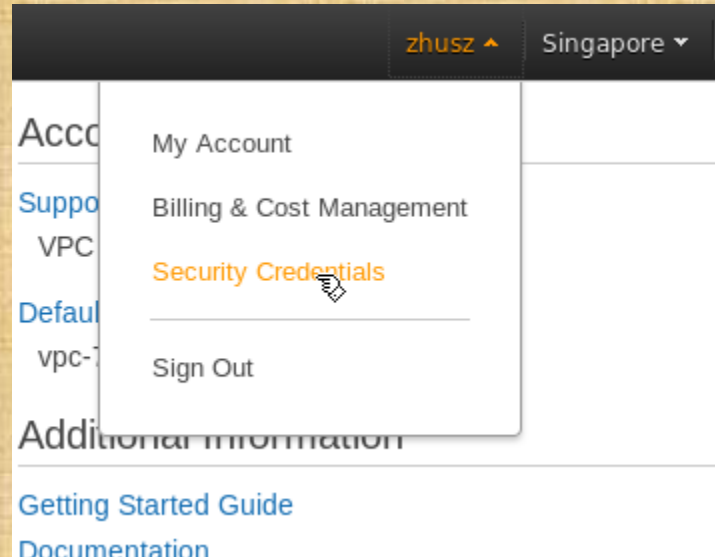


Step 3: Make your project in remote instance

- In the virtual env, we have installed the eb package. Now we will use it.
- Check if it exist: `eb --version`
- Using elastic beanstalk, we would create and connect to an instance in a different way with last time.
- What is needed for this time based on your Tutorial 3 task:
 - Your aws account;
 - Your security group;
 - The key pair for your ssh connection;
 - Your security credential (I did not cover this last time, so I would first mention this.)

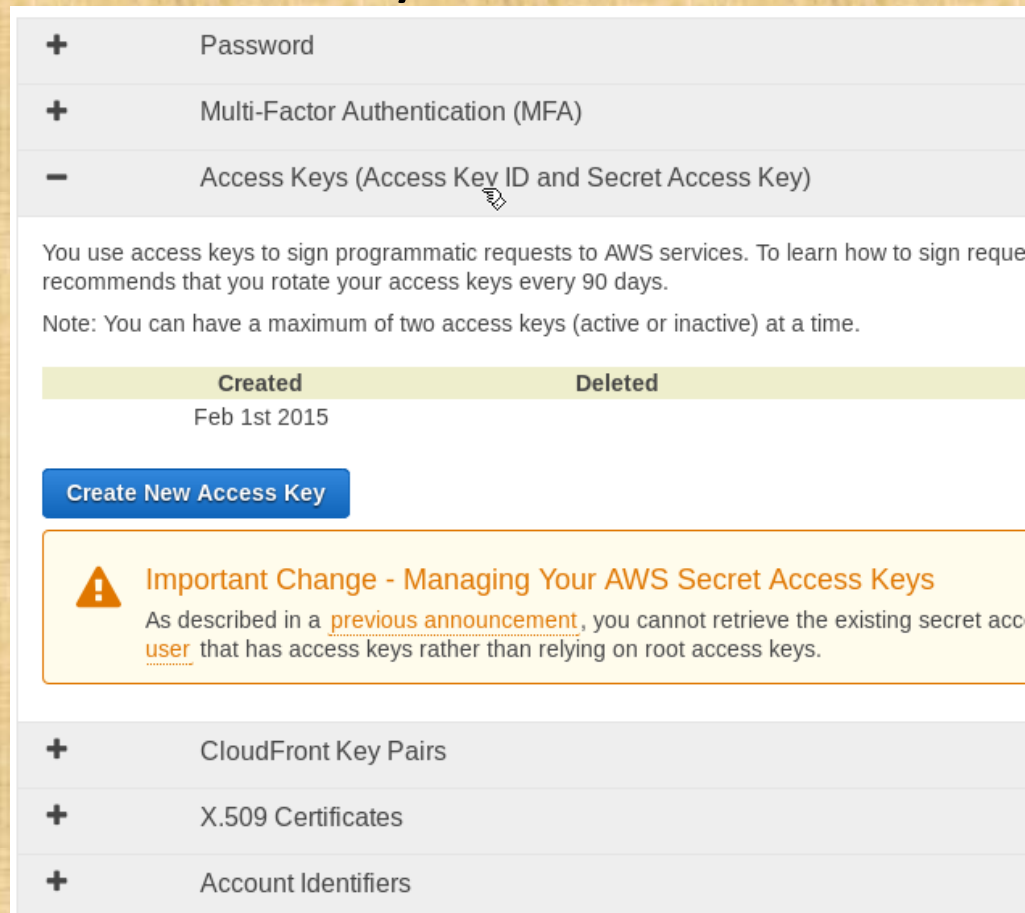
Step 3: Create your security credential

- Logging into your AWS account, choose security credential



Step 3: Create your security credential

- Inside, choose the access keys



The screenshot shows the 'Security credentials' page in the AWS IAM console. At the top, there are three expandable sections: 'Password', 'Multi-Factor Authentication (MFA)', and 'Access Keys (Access Key ID and Secret Access Key)'. The 'Access Keys' section is expanded, showing a table with columns 'Created' and 'Deleted'. The 'Created' column contains the date 'Feb 1st 2015'. Below the table is a blue button labeled 'Create New Access Key'. A yellow warning box with an exclamation mark icon contains the text: 'Important Change - Managing Your AWS Secret Access Keys. As described in a [previous announcement](#), you cannot retrieve the existing secret access key for a user that has access keys rather than relying on root access keys.' Below the warning box, there are three more expandable sections: 'CloudFront Key Pairs', 'X.509 Certificates', and 'Account Identifiers'.

+	Password
+	Multi-Factor Authentication (MFA)
-	Access Keys (Access Key ID and Secret Access Key)

You use access keys to sign programmatic requests to AWS services. To learn how to sign requests, see [Using the AWS CLI](#). AWS recommends that you rotate your access keys every 90 days.

Note: You can have a maximum of two access keys (active or inactive) at a time.

Created	Deleted
Feb 1st 2015	

[Create New Access Key](#)

Important Change - Managing Your AWS Secret Access Keys

As described in a [previous announcement](#), you cannot retrieve the existing secret access key for a [user](#) that has access keys rather than relying on root access keys.

+	CloudFront Key Pairs
+	X.509 Certificates
+	Account Identifiers

Step 3: Create your security credential

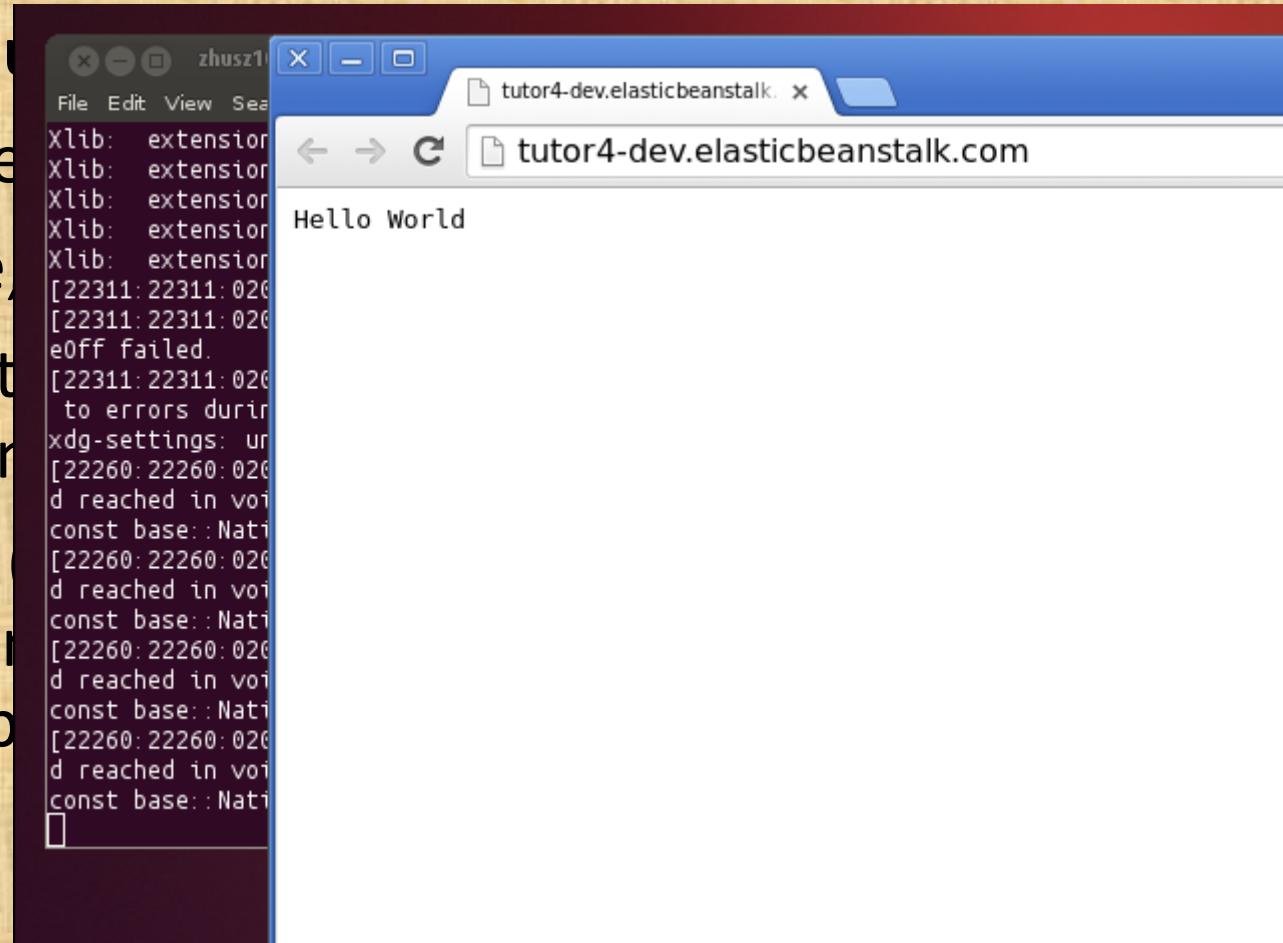
- In your case, if you have never created any before, there should be no items there.
- Click create new access key, the credential id and private key serial. Save the csv file in safe place so that you can refer to it later.
- When you do the eb init, you need to input the information.
- If you lose the private key (which would be no longer retrieved in the future), you will no longer connect to your previous instance!!! DO KEEP YOUR PRIVATE KEY SAFELY!!!
- Please also put your previously created private key pair to the ~/.ssh directory, so you can connect to your instance under eb method.

Step 3: Launch and connect to your instance

- Now we are under eb.
- (1) eb init, need your security confidential;
- (2) eb create, launch new instance;
- Above two steps needs some interactive input, don't forget to choose Singapore, and input your instance name appropriately.
- (3) eb open, (please exe this command under terminal in graphical interface). Since we have put the project (verified in Step 2), now our hello world page can be viewed globally.

Step 3: Launch and connect to your instance

- Now we are
- (1) eb init, ne
- (2) eb create,
- Above two st
- Singapore, ar
- (3) eb open, (
- interface). Si
- hello world p

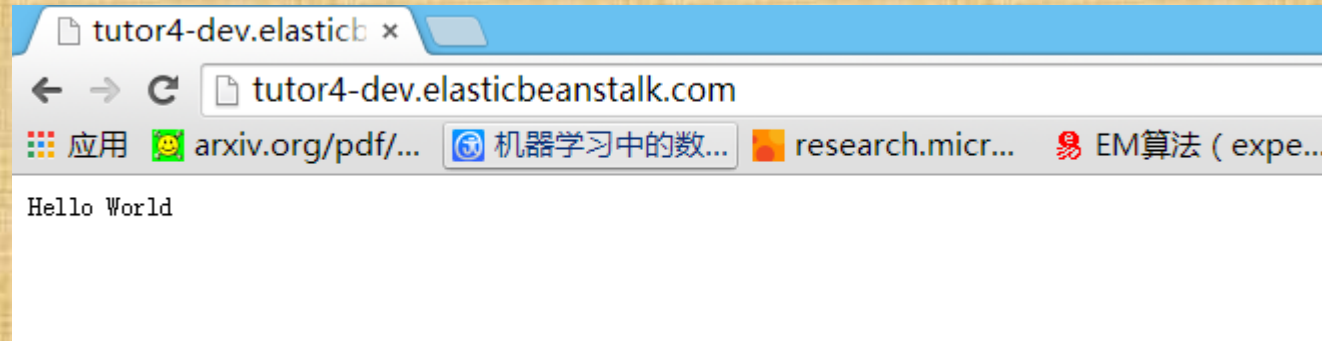


target to choose

in graphical
p 2), now our





Step 3: Launch and connect to your instance

- It can also be viewed via another machine



Step 4: Development locally and remotely

- Now you can develop your web page both locally and remotely.
- Every time you make some update, first git commit, then use eb deploy. You can see new features added when you open the public url.
- Use eb ssh to ssh connect to your instance. This requires that the private key pair (*.pem) should be located under ~/.ssh and with 600 accessibility.
- In fact this instance is similar to that we created using UI interface.

Filter by tags and attributes or search by keyword						
<input type="checkbox"/>	Name ▾	Instance ID ▲	Instance Type ▾	Availability Zone ▾	Instance State ▾	Status Checks ▾
<input type="checkbox"/>		i-749e19b9	t2.micro	ap-southeast-1b	 running	 2/2 checks...
<input type="checkbox"/>	tutor4-dev	i-fc9a1d31	t1.micro	ap-southeast-1b	 running	 2/2 checks...

- Thank you.