Server Setup Whitepaper

Process 1: Launching an instance

1. Log into AWS console (select “Sign In to the Console” button on https://aws.amazon.com/)
2. Select “EC2” button. On dashboard, select “Instances” link or “X Running Instances”, to view your existing instances.
3. To deploy a new server, select “Launch Instance”
4. On Step 1 (Choose an AMI), select “My AMIs” and select the latest “arcane standard” version.
5. On Step 2 (instance type), select the size of VM required and then click “Review and Launch”
6. This will skip to Step 7.
    a. Select “Edit security groups”
    b. Click “select an existing security group” and in the checkboxes below, select all security groups you would like to associate. This is a firewall but does not control the server’s IPTables which are an added layer. The standard is the “arcane-lamp-image-webserver…” group.
    c. Click “Review and Launch”
7. Select “Launch” (you may configure storage and tags as you see fit, before launching).
    a. In the popup, select “choose an existing key pair” from the 1st dropdown.
    b. Select the “arcane-lamp-image” key in the 2nd dropdown.
    c. Select the “I acknowledge” checkbox and click “Launch instances”.
8. You are now done creating the server, it may take a few minutes to initialize. Click to view your instances (or select “Instances” on the left menu) and review the “instance state” and “status checks” column. You should have a “running” state with 2/2 checks passing (green checkmark). The GUI is often slow to update, so refreshing the page every couple minutes might yield positive results. Once initialized, continue to “connecting”, below.
9. It’s best for you to assign a name in the 1st column, as this will make identification easier.

Process 2: Connecting to your instance

1. Select your new instance. In the “description” pane, below, you will see the Public DNS and “Public IP” that you will use to connect.
    a. **NOTE:** Your server does not yet have an elastic IP, the “public IP” column will be black text to indicate that (blue indicates an elastic IP). You may also verify this by selecting “Elastic IP’s” to view existing IP’s. Step 3 reviews Elastic IP’s further. If you do not have an elastic IP, rebooting your server may change your public IP, so it is advised that you use the “Public DNS” address until an elastic IP is assigned.
2. In your putty or SSH console, connect to the Public DNS using Arcane’s default PEM key file:
    a. $ssh -i "Path/to/your/arcane_lamp_image.pem" ubuntu@ec2-50-112-216-140.us-west-2.compute.amazonaws.com

ARCANE STRATEGIES

   i. You'll notice the user is "ubuntu". The "root" user will also work but will request you to switch to "ubuntu". Ubuntu is a sudoer, so you'll use "sudo" for most of these commands or you may switch user to root with "su -"

  b. You may also choose to store this key on your local putty or console key repository (ie. your /.ssh keys or Putty Pageant). Linux instances or PuttyGen may be used to convert the PEM file to a usable PPK format. This will allow you to connect without explicitly identifying the pem file
     i. $ssh ubuntu@ec2-50-112-216-140.us-west-2.compute.amazonaws.com
  c. If you have an additional key that you would like to include, you may include that on a new line in the server's key file: ~./ssh/authorized_keys

3. To enable an elastic IP, select "Elastic IP" and click the "Allocate New Address" button.
   a. When prompted, select "Yes, Allocate" and "Close"
   b. Now select your Elastic IP that you've just created. The prompt will have told you this but you'll also note that it's new by the fact that it has no public DNs nor instance, identified.
   c. Click "Actions" > "Associate Address"
   d. Select the "Instance" field to get a dropdown of your instances and select it or type until you find the one you're looking for. This is where your instance name will come in handy, otherwise you'll need to reference the ID in the "instances" tab (in another window).
   e. Once selected, click "associate". Don't worry about the private and reassociation fields for this standard process, those are for specific cases (you may click the "i" for more info).
   f. Now you may use this IP for DNS association and SSH connection methods, instead of the Public DNS.
4. To enable SFTP password authentication login for clients or other access, perform the following:
   a. $sudo vi /etc/ssh/sshd_config
   b. ## You may use vim or nano if you choose. To insert with vi, select "i". To close and save, select "ctrl+[" and ":wq" // if you're overwriting a swap file, use ":wq!" to force the action.
   c. ## Find "PasswordAuthentication" and change "no" to "yes".
   d. ## By default, your image will come with users "client", "ftpuser", and "arcane". Credentials for each are available with your administrator. MySQL credentials are also available with your administrator and include a "root" user.
5. You are now connected. If you are unable to connect due to firewall restrictions, you should check the security group to be sure that standard protocols (SSH, MySQL, HTTP) are opened to your local IP address/range. The server also has its own IPTables controlled in /etc/iptables/rules.v4, which may require editing. Should this be necessary and you are unable to edit them, you may work around this by detaching the volume of your new server, attaching it and mounting it as a secondary partition, to a clean instance that you are able to connect to, and editing it there. Then stop the instance, detach the volume, and reattach it as the primary drive/partition for your instance.

Process 3: Creating and Enabling a Site’s Vhost for a migration

1. $sudo mv /etc/apache2/sites-available/yourdomain.conf /etc/apache2/sites-available/[yournewdomain].conf
   a. ## This copies the default vhost config that’s already set up for you
2. $sudo a2dissite yourdomain
   a. ## Disable the default vhost that we no longer need.
3. $sudo mv /var/www/html/yourdomain /var/www/html/[yournewdomain]
   a. ## Rename the default webroot and its contained log files.
   b. ## If you have multiple sites: $sudo cp -R /var/www/html/[yournewdomain] /var/www/html/[your2nddomain]
4. $sudo vi /etc/apache2/sites-available/[yournewdomain].conf
   a. ## Open your vhost config for editing. You may use vim or nano, as well.
   b. ## Set directory paths correctly. ServerName should be www version, alias will be non-www version.
5. $sudo service apache2 reload
   a. ## If apache2 isn't running on boot:
      i. $sudo service apache2 start
      ii. $sudo update-rc.d apache2 enable
      iii. ## Sets apache2 to run on boot
6. $sudo a2ensite [yournewdomain]
   a. Enables your new domain
7. $sudo service apache2 reload
   a. Completes the process
8. Now you can import your files:
   a. $cd /var/www/html/[yournewdomain]
   b. $git init
      i. By default you should already have an empty git repo created, so this should not be necessary.
   c. $git clone [your report repository]
      i. You will be asked to log into your account.
   d. $touch .gitignore
      i. It’s advised that you create a gitignore file to prevent overwriting config files before you create them. To learn more about how to add them now and in the future, see https://git-scm.com/docs/gitignore
      ii. Now create or edit any files that might otherwise not be contained in your repository such as config files or .htaccess files. “Touch” creates new files, vi/vim/nano edits them.
9. Now you can import your database and set up new accounts.
   a. $mysql -u root -p
      i. You will be prompted to login. Your administrator has the default root pw.
   b. mysql > CREATE DATABASE ‘yournewdatabase’;
   c. mysql > CREATE USER 'newuser'@'localhost' IDENTIFIED BY 'password';
   d. mysql > GRANT ALL PRIVILEGES ON ‘yournewdatabase’. * TO 'newuser'@'localhost';
      i. ‘yournewdatabase’.* corresponds to database.table. This gives your new user

full permissions over all tables that might exist in 'yournewdatabase'. If you must have this user have full root level permissions over all databases, simply use "*.*".

e. mysql> FLUSH PRIVILEGES;
f. mysql> exit;
g. $mysql -u 'newuser' -p 'yournewdatabase' < /path/to/your/databasefile.sql
    i. You will be prompted for your password. This is the one identified in step 9.c above.
    ii. You must upload your database file to import it. You may choose to make it publicly accessible and download with wget but it's better to upload it via SFTP (see connecting to your instance, above), or find it in your webroot, as it may have been stored in your repository.
h. Now update your config string in your application and you're done!

10. Change your DNS and you're done!