Warszawskie Forum Julia
Julia Warsaw Meetup
January 10th, 2018

How to build your Julia computational environment in the AWS cloud

Przemysław Szufel, PhD
https://szufel.pl/

SGH Warsaw School of Economics

Why the cloud?

- Shorter times to deliver results
  - Dynamically adjust the size of resources to the size of a problem
  - Cost of 10 instances for 10h = cost of 400 instances for 15 minutes
- Cost efficient with AWS EC2 spot
  - 1000 vCPU cores can be bought from $7 an hour
  - Spot-fleet functionality to manage cost-optimal computations at scale
- Solutions for computational clusters
  - KissCluster – lightweight cluster in 2 minutes
  - CfnCluster – full-blown numerical computing solution
- Cheap and data storage with fast access – S3 (0.023$/GB-month)

Agenda

- AWS EC2 options for a computational scientists
- Create an EC2 virtual machine with Julia
- How to build your own Julia machines
- Configure IDE on the cloud
  - Julia Jupyter notebook in headless environments
  - Cloud9 with Julia
- Configure data storage on AWS S3

AWS EC2 instance types for a computational scientist

- Family t2.*
  - Cheapest, t2.micro free for one year
  - Good for testing and explorative usage
- Family c4.*, c5.*
  - Computational power oriented
  - Applications : simulations, numerical computing
- Family r4.*, x1.*
  - RAM memory oriented (RAM up to 4TB)
  - Applications with in-memory analytics
- Family m4.*, m5.* – intermediate between c4.* and r4.*
- Family p2.*, p3.*, g2.*
  - GPU computing, (g2.* more graphics oriented)
  - Applications : deep learning

Preinstalled machine - AMIs

- An Amazon Machine Image (AMI) provides the information required to launch an instance, which is a virtual server in the cloud. You specify an AMI when you launch an instance, and you can launch as many instances from the AMI as you need. You can also launch instances from as many different AMIs as you need.
- An AMI includes the following:
  - A template for the root volume for the instance (for example, an operating system, an application server, and applications)
  - Launch permissions that control which AWS accounts can use the AMI to launch instances
  - A block device mapping that specifies the volumes to attach to the instance when it's launched

Source: http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html

Spot auction market

- 80-90% cheaper than regular prices
  - 1000vCPUs can be bought for 7$ an hour
- Sold through clearing price auction mechanism
- Spot instances run when your bid price exceeds the Spot price
- Spot Prices continuously change independently:
  - in each 36 data centers across 14 geographic regions
  - for each of 90 server types
- In December 2017 Amazon has flattened spot prices to make the predictable

Buy computing resources cheaper with spot

Simulation Modelling Practice and Theory
ELSEVIER Volume 58, Part 2, November 2015, Pages 172-187

On optimization of simulation execution on Amazon EC2 spot market

Bogumił Kamiński, Przemysław Szufel

Show more

https://doi.org/10.1016/j.simpat.2015.05.008

Get rights and content

Obtaining Julia for the AWS

- Install from binaries

- Build it yourself
  → Beware of incompatibility between different EC2 instance types!

- Use JuliaPro AMIs
  → Simplest ☺

Installing Julia from binaries

```
sudo add-apt-repository ppa:staticfloat/juliareleases
sudo apt update
sudo apt install --yes build-essential
sudo apt upgrade
sudo apt-cache show julia
sudo apt install --yes julia julia-doc –y
```

→ The binary is usually updated with some delay compared to source code version

All the examples have been tested on Ubuntu 16

Julia Build it yourself
(1) – prepare the environment

```
sudo apt-get update
sudo apt-get install --yes build-essential python-minimal gfortran m4 cmake pkg-config libssl-dev htop
git clone git://github.com/JuliaLang/julia.git
cd julia
git checkout v0.6.2
```

All the examples have been tested on Ubuntu 16

Julia Build it yourself (2) – decide

- Option (1) - Build Julia with Open BLAS an Open LIBM

```
make -j $((`nproc`-1)) 1>build_log.txt 2>build_error.txt
```

- Option (2) - Build Julia with Intel MKL an Open LIBM

```
echo "USE_INTEL_MKL = 1" >> Make.user
source /opt/intel/bin/compilervars.sh intel64
make -j $((`nproc`-1)) 1>build_log.txt 2>build_error.txt
```

- Option (3) Build Julia with Intel MKL and Intel LIBM

```
echo "USE_INTEL_MKL = 1" >> Make.user
echo "USE_INTEL_LIBM = 1" >> Make.user
source /opt/intel/bin/compilervars.sh intel64
make -j $((`nproc`-1)) 1>build_log.txt 2>build_error.txt
```

IDE options for Julia in the cloud

- Terminal (vim, emacs, nano, etc…)

- Jupyter notebook
  - → has syntax support
  - → remember that you are running in headless mode

- Cloud9
  - → currently no syntax support except for highlighting
  - → collaborative code editing, collaborative Julia REPL
  - → works with networks that block SSH connections

Jupyter notebook in headless mode…

```
sudo ln -s /home/ubuntu/julia_path/julia /usr/local/bin/julia
ssh -i keyfile.pem -L 8888:127.0.0.1:8888 ubuntu@ec2-18-217-153-198.us-east-2.compute.amazonaws.com
```

Now run either of the commands (depending on the Jupyter location):

- `.local/bin/jupyter notebook`
- `~/.julia/v0.6/Conda/deps/usr/bin/jupyter notebook`

or from Julia console

```
using IJulia
notebook(detached=true)
run(`$(IJulia.notebook_cmd[1]) notebook list`)
```

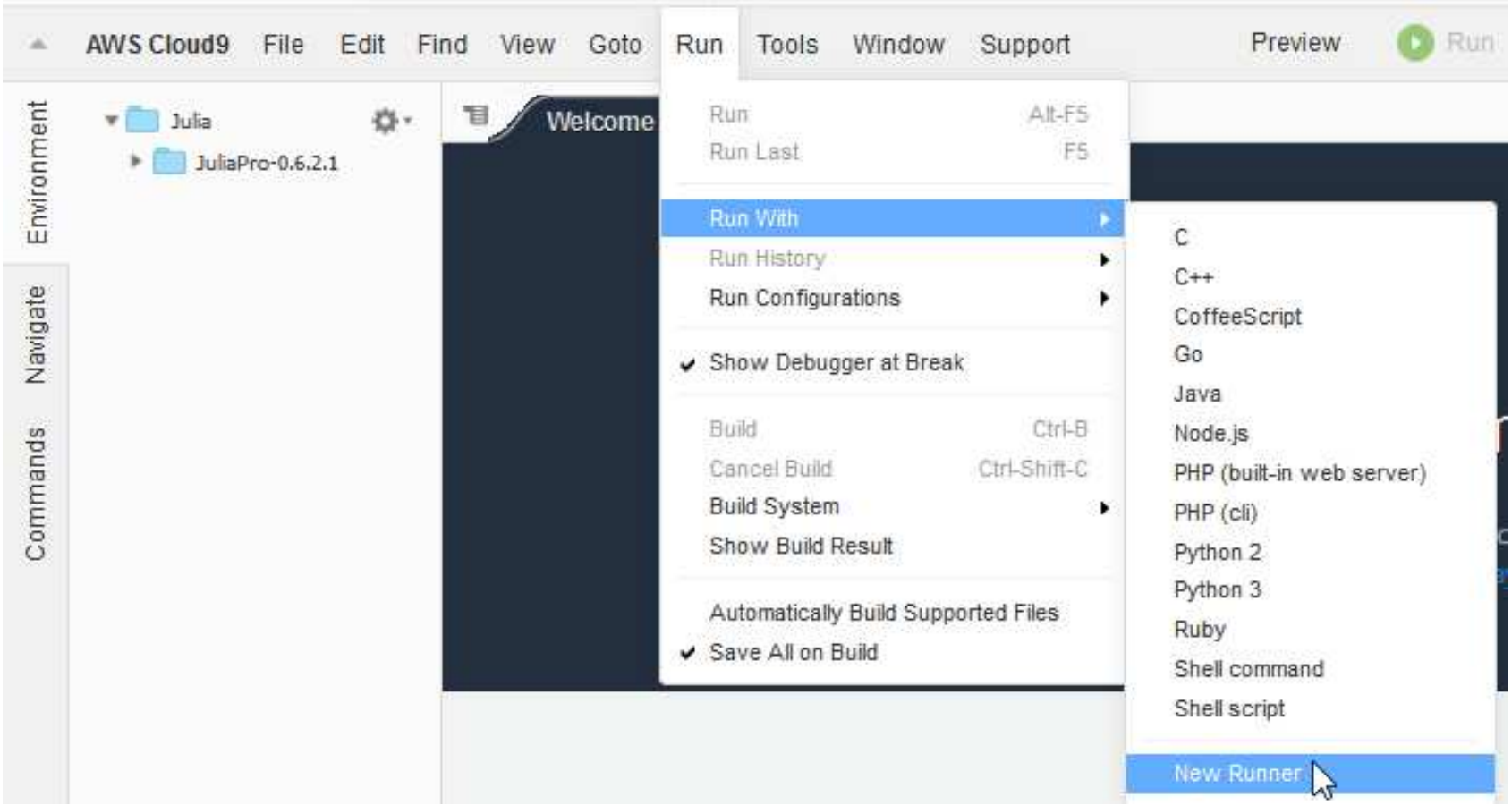
Setting up Cloud9 on AWS for Julia

1. Prepare a Linux machine with installed Julia
2. Install node.js

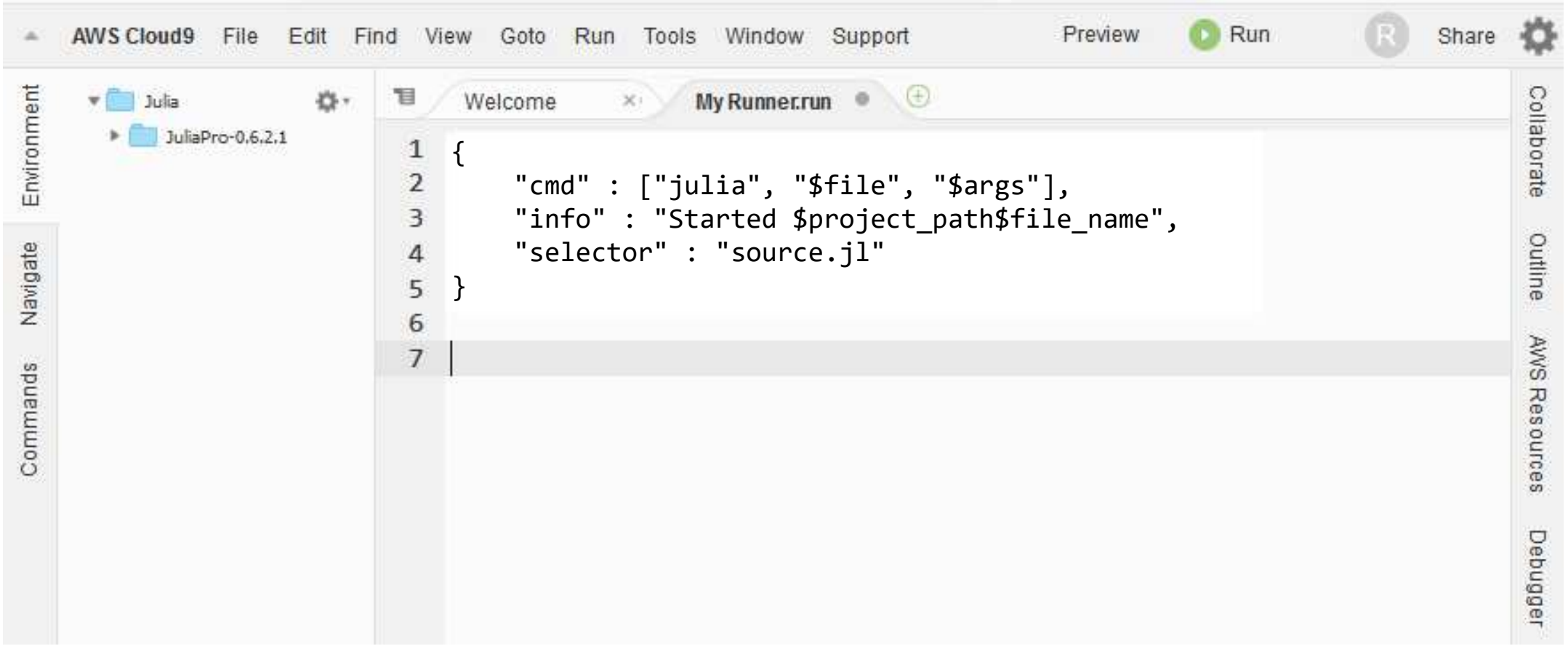
```
sudo apt update
sudo apt install nodejs-legacy
```

3. Make sure that your server is accepting external SSH connections
   - configure instance Security Group to accept SSH connections from 0.0.0.0/0
4. In the AWS console go to Cloud9 service and start creating a new environment.
   1. Select the "Connect and run in remote server (SSH)" option
   2. user name: ubuntu
   3. Provide the host name of your EC2 instance
5. Configure SSH authorization.
   1. "Environment settings" → "Copy key to clipboard" to copy the key
   2. Open in terminal an SSH connection to your remote server.
   3. `echo [paste-key-here] >> ~/.ssh/authorized_keys`

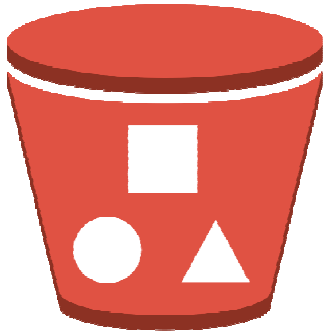
Configuring Cloud9 Runner for Julia (1)



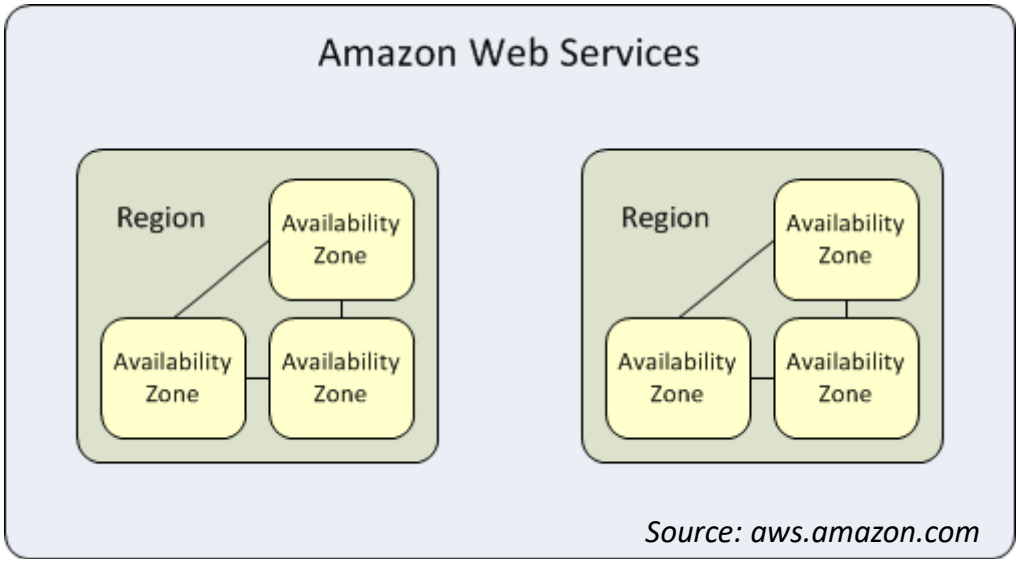
Configuring Cloud9 Runner for Julia (2)



S3 containers (=buckets)



- Place to keep all your data in the cloud
- Data assigned to a region (e.g. Ohio)
- 99.999999999% durability (probability of keeping a file throughout a single year)
- 99.99% availability (time the file can be accessed)
- Access types
  - bash
  - Directly from Julia



S3 container access methods

- EC2 instance assumes a role
- The role has a policy attached to it

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [      "s3:ListBucket"       ],
            "Resource": [ "arn:aws:s3:::szufel-julia"   ]
        },
        {
            "Effect": "Allow",
            "Action": [ "s3:PutObject", "s3:GetObject", "s3:DeleteObject"],
            "Resource": ["arn:aws:s3:::szufel-julia/*"           ]
        }
    ]
}
```

Accessing data via awscli

- Installation

```
sudo apt update
sudo apt install awscli
```

- Usage
  - List files:

```
aws --region us-east-2 s3 ls
```

  - Send file

```
aws --region us-east-2 s3 cp local_file s3://bucket_name/remote-file
```

  - Get file to a local folder – do not forget the dot . at the end !

```
aws --region us-east-2 s3 cp s3://bucket_name/remote-file .
```

Conclusions

- Powerful computing environment that scales to your needs
  - Select hardware
  - Change the hardware in few minutes
- Infinite, cheap data storage with S3
- Create your own AMI and use it in the future or share with others
- Collaborative work possible with Cloud9
- Cloud9 on EC2 is free (you pay only for the EC2 instance)