Unit 4 Activity:

```
echo "Starting OpenStack provisioning..."

echo "Creating instance: $INSTANCE_NAME"
openstack server create \
  --flavor $FLAVOR \
  --image "$IMAGE" \
  --nic net-id=$(openstack network show $NETWORK -c id -f value) \
  --security-group $SEC_GROUP \
  --key-name $KEYPAIR \
  --user-data $USERDATA \
  $INSTANCE_NAME

echo "Waiting for instance to be ACTIVE..."
status=""
while [[ $status != "ACTIVE" ]]; do
    status=$(openstack server show $INSTANCE_NAME -c status -f value)
    echo "Current status: $status"
    sleep 5
done
echo "Instance $INSTANCE_NAME is ACTIVE."

IP=$(openstack server show $INSTANCE_NAME -c addresses -f value | awk -F'=' '{print $2}')
echo "Instance IP: $IP"

echo "Waiting for instance to initialize..."
sleep 30
echo "Running basic commands on instance..."
ssh -o StrictHostKeyChecking=no ubuntu@$IP "uname -a && df -h && uptime"

echo "Automation finished successfully."
```

How Automation Using Scripting Helps in Cloud Operations

Cloud platforms like OpenStack allow us to create virtual machines, networks, and storage on demand. Doing this manually through a dashboard can be slow and repetitive. Automation through scripting solves this problem by turning these steps into commands that run in sequence without manual input.

In general, automation makes the process **faster**, creating an instance normally requires several clicks and waiting between steps. A script can handle everything automatically, including checking if the instance is ready before continuing. This saves time and reduces human error.

Second, automation ensures **consistency**. If we set up multiple servers manually, small mistakes can happen, such as forgetting to install a package or applying the wrong security group. A script follows the same steps every time, so the setup is always correct.

Third, automation helps with **scaling**. In a real business environment, we might need to create ten or even hundreds of servers for an application. Doing that manually would be impossible. With a script, the same code can be reused to create as many servers as needed.

The script above uses the OpenStack command-line tool. It defines variables like the instance name, image, flavor, network, and key pair. Then it runs openstack server create to start the instance. After that, it checks the status every few seconds until it becomes “ACTIVE.” Once the instance is running, it gets the IP address and uses SSH to run basic commands like uname -a (to check the OS), df -h (to check disk space), and uptime.
We also use a cloud-init script called userdata.sh. This is passed when the instance is created, so it automatically installs Apache and sets up a simple webpage. This means that after the script finishes, the instance is already configured without needing to log in manually.

In a real cloud environment, such scripts can be part of larger workflows, such as auto-scaling or disaster recovery. They can also be combined with CI/CD pipelines to deploy applications quickly. Automation helps reduce costs by shutting down unused instances automatically and improves security by applying predefined security settings.

To sum it up, automation with Bash scripts is a simple but powerful way to manage cloud infrastructure. It reduces manual work, prevents errors, and enables scalability.