Day 3: LV

- :
  - Linear LV
  - Striped LVMirrored LV
  - `fio`
- :
  -
  - **Go `lvm-manager` LV Go**

(30-40%)

- :

  1. **(Logical Volume - LV)**: LVM LV VG ""PEs LV `/dev/vg_data_01/lv_web` `/dev/sda1`

- :

  1. **(Linear LV)**: LV LVM VG PV PE PV PE PV ""
  2. **(Striped LV)**:
     - : PVChunk""Stripe PV 1MB 64KB PV1 64KB PV2 64KB PV1
     - : I/O
     - : PV LV
  3. **(Mirrored LV)**:
     - : 2PV LVM
     - : PV PV
     - : **50%** 1TB 2TB

- :

  - :
  - :
  - :

□ □□□□ (40-50%)

Day 2 `vg_data_01` `/dev/sdb`, `/dev/sdc` `/dev/sdd`, `/dev/sde`

: `/dev/sdd` `/dev/sde` PV VG

```
sudo pvcreate /dev/sdd /dev/sde
#  VG
sudo vgcreate vg_safe_01 /dev/sdd /dev/sde
```

1.

```
# 1.  vg_data_01  2GB  LV lv_linear_data
sudo lvcreate -L 2G -n lv_linear_data vg_data_01

# 2.  ext4
sudo mkfs.ext4 /dev/vg_data_01/lv_linear_data

# 3.
sudo mkdir -p /mnt/linear_data
sudo mount /dev/vg_data_01/lv_linear_data /mnt/linear_data

# 4.
df -hT /mnt/linear_data
#  2G ext4
```

2.

`vg_data_01` PV

```
# 1.  4GB (Stripe Size) 64KB
# -i 2:  2  PV  (stripes)
# -I 64:  64KB (Stripe Size)
sudo lvcreate -L 4G -i 2 -I 64 -n lv_striped_data vg_data_01

# 2.
sudo mkfs.ext4 /dev/vg_data_01/lv_striped_data
sudo mkdir -p /mnt/striped_data
sudo mount /dev/vg_data_01/lv_striped_data /mnt/striped_data

# 3.
df -hT /mnt/striped_data
```

3.

`vg_safe_01`

```
# 1.  2GB
# -m 1:  1  2  PV
# --mirrorlog core:
sudo lvcreate -L 2G -m 1 --mirrorlog core -n lv_mirrored_data vg_safe_01

# 2.
sudo mkfs.ext4 /dev/vg_safe_01/lv_mirrored_data
sudo mkdir -p /mnt/mirrored_data
sudo mount /dev/vg_safe_01/lv_mirrored_data /mnt/mirrored_data

# 3.
df -hT /mnt/mirrored_data
#  LV
sudo lvs -o +devices vg_safe_01
#  lv_mirrored_data
```

4.

`fio`

```
# 1.  fio
# CentOS/RHEL: sudo dnf install -y fio
# Ubuntu/Debian: sudo apt-get install -y fio

# 2.
sudo fio --name=linear_write --directory=/mnt/linear_data --size=500M --direct=1 --rw=write --bs=1M --ioengine=libaio --runtime=20 --group_reporting

# 3.
sudo fio --name=striped_write --directory=/mnt/striped_data --size=500M --direct=1 --rw=write --bs=1M --ioengine=libaio --runtime=20 --group_reporting

# 4.
#  fio  bw (Bandwidth)
```

Go (20-30%)

: lvm-manager LV LV

:

```
#  lvm-manager
mkdir -p cmd/day03
cd cmd/day03
```

(main.go):

```
package main

import (
        "bytes"
        "encoding/json"
        "fmt"
        "log"
        "os/exec"
        "strings"
)

// ---  Day 2  ---
type LVMReport struct {
        Report []map[string][]map[string]string `json:"report"`
}
// ... ( Day 2  PhysicalVolume, VolumeGroup  runLVMCommand, GetPhysicalVolumes, GetVolumeGroups )

// ---  LogicalVolume  ---
type LogicalVolume struct {
        Name   string `json:"lv_name"`
        VG     string `json:"vg_name"`
        Attr   string `json:"lv_attr"`
        Size   string `json:"lv_size"`
        Origin string `json:"origin"` // For snapshots
        Path   string `json:"lv_path"`
}

// ---  GetLogicalVolumes  ---
func GetLogicalVolumes() ([]LogicalVolume, error) {
        // -o  lv_attr,lv_path
        output, err := runLVMCommand("lvs", "-o", "lv_name,vg_name,lv_attr,lv_size,origin,lv_path")
        if err != nil {
                return nil, err
        }

        var report LVMReport
        if err := json.Unmarshal(output, &report); err != nil {
                return nil, fmt.Errorf("failed to parse lvs JSON: %v", err)
        }

        var lvs []LogicalVolume
        if len(report.Report) > 0 && report.Report[0]["lv"] != nil {
                for _, lvMap := range report.Report[0]["lv"] {
                        lvs = append(lvs, LogicalVolume{
                                Name:   lvMap["lv_name"],
                                VG:     lvMap["vg_name"],
                                Attr:   lvMap["lv_attr"],
                                Size:   lvMap["lv_size"],
                                Origin: lvMap["origin"],
                                Path:   lvMap["lv_path"],
                        })
                }
        }
        return lvs, nil
}

// ---  CreateLinearLV  ---
// CreateLinearLV creates a standard linear logical volume.
// size is in Gigabytes (G).
func CreateLinearLV(vgName, lvName string, sizeG int) error {
        sizeStr := fmt.Sprintf("%dG", sizeG)
        log.Printf("Attempting to create LV: Name=%s, VG=%s, Size=%s", lvName, vgName, sizeStr)

        //  -L -n
        _, err := runLVMCommand("lvcreate", "-L", sizeStr, "-n", lvName, vgName)
        if err != nil {
                return fmt.Errorf("failed to create linear LV %s in VG %s: %w", lvName, vgName, err)
        }

        log.Printf("Successfully created LV %s.", lvName)
        return nil
}

// runLVMCommand ( Day 2 )
func runLVMCommand(command string, args ...string) ([]byte, error) {
        fullArgs := append([]string{command}, args...)
        // lvcreate  reportformat json,
        if command != "lvcreate" {
            fullArgs = append(fullArgs, "--reportformat", "json")
    }
        cmd := exec.Command("sudo", fullArgs...)
        var stdout, stderr bytes.Buffer
```

```
        cmd.Stdout = &stdout
        cmd.Stderr = &stderr
        err := cmd.Run()
        if err != nil {
                return nil, fmt.Errorf("command `sudo %s %s` failed: %v\nStderr: %s", command, strings.Join(args, " "), err, stderr.String())
        }
        return stdout.Bytes(), nil
}


func main() {
        log.Println("--- Phase 1: Creating a new LV with Go ---")
        //  LV
        err := CreateLinearLV("vg_data_01", "lv_from_go", 1)
        if err != nil {
                log.Printf("WARN: Could not create lv_from_go: %v. It might already exist.", err)
        }

        log.Println("\n--- Phase 2: Fetching LVM logical volumes ---")
        lvs, err := GetLogicalVolumes()
        if err != nil {
                log.Fatalf("Error getting logical volumes: %v", err)
        }

        fmt.Println("\n--- Logical Volumes (LVs) ---")
        fmt.Printf("%-20s %-15s %-12s %-10s %-s\n", "LV Path", "VG Name", "Attributes", "Size", "Origin")
        fmt.Println(strings.Repeat("-", 80))
        for _, lv := range lvs {
                origin := lv.Origin
                if origin == "" {
                        origin = "-"
                }
                fmt.Printf("%-20s %-15s %-12s %-10s %-s\n", lv.Path, lv.VG, lv.Attr, lv.Size, origin)
        }
        log.Println("LVM information fetched successfully.")
}
```

- :
  - : `lvcreate` "Volume group "vg_data_01" has insufficient free space"
    - : PE LV `vgs` `vgdisplay` VFree
    - : LV `vgextend` VG PV
  - : `lvcreate -i 2` "Cannot create striped LV with only 1 PVs"
    - : N N PV
    - : VG PV
- :
  - **(Stripe Size)**: I/O 16K 32K I/O 256K 512K
- :
  - : LV I/O Web LV
  - **(Alignment)**: LVM PVLV I/O

- : Go `CreateLV`
- : `func CreateLV(vgName, lvName string, sizeG int, lvType string, stripes int) error`
  - `lvType` "linear", "striped"
  - `lvType` "striped" `stripes`
  - `lvType` `lvcreate`
  - `GetVolumeGroups` `GetPhysicalVolumes`
    - VG
    - VG
    - VG PV
  -

- :
  1. :
     - `lvchange vg_safe_01` PV `/dev/sdd`
     - `lvs -a -o +devices` "degraded"
     - `/mnt/mirrored_data`
     - `vgchange lvconvert --repair`
  2. **Go** :
     - `lvm-manager list` `list pv`, `vg`, `lv` `go run main.go list lv` LV
     - : `os.Args` flag