Day 5: LVM Go

- : LVM
- : ext4 xfs
- **Go**: Go
- : LVM
- : Go LVM LVM LVM

(30%)

1.

Online ResizingLVM

- **LVM** : `lvextend` LVVGPE PE LV LVM LV
- : `resize2fs` (for ext4) `xfs_growfs` (for XFS)

2.

- :
  - **VG** : VG
  - : LV
  - :
- :
  1. : LVM (`vgcfgbackup`)
  2. : VG (`vgdisplay`)
  3. **LV**: `lvextend`
  4. : () `e2fsck -f` (ext4)
  5. : `resize2fs` `xfs_growfs`
  6. : `df -h`

🔲 🔲🔲🔲🔲 (40%)

 `data_lv` `/data` VG `storage_vg`

1.

```
# 
sudo vgdisplay storage_vg | grep "Free  PE"

# 
df -hT /data
```

2. (lvextend)

```
#  data_lv  2GB
sudo lvextend -L +2G /dev/storage_vg/data_lv

#  10GB
# sudo lvextend -L 10G /dev/storage_vg/data_lv

#  LV
sudo lvdisplay /dev/storage_vg/data_lv
```

: `lvextend` "Size of logical volume ... changed from X to Y. Logical volume ... successfully resized."`lvdisplay` LV Size `df -h`

3.

ext4

```
# 
sudo e2fsck -f /dev/storage_vg/data_lv

# 
sudo resize2fs /dev/storage_vg/data_lv
```

: `resize2fs`

XFS

XFS `xfs_growfs`

```
# XFS
sudo xfs_growfs /data
```

: `xfs_growfs`

4.

```
# 
df -hT /data
```

:df -h

Go (30%)

lvm-autoscaler LV

1.

```
lvm-manager/
├── cmd/
│   └── main.go
├── internal/
│   ├── lvm/
│   │   └── lvm.go
│   └── monitor/
│       └── monitor.go
├── pkg/
│   └── utils/
│       └── exec.go
└── configs/
    └── config.yaml
```

2. `configs/config.yaml`

```
monitor:
  interval_seconds: 60
  targets:
    - lv_path: "/dev/storage_vg/data_lv"
      mount_point: "/data"
      threshold_percent: 80
      increment_gb: 2
```

3. `pkg/utils/exec.go`

```
package utils

import (
        "bytes"
        "os/exec"
        "strings"
)

// RunCommand  shell
func RunCommand(name string, args ...string) (string, error) {
        cmd := exec.Command(name, args...)
        var stdout, stderr bytes.Buffer
        cmd.Stdout = &stdout
        cmd.Stderr = &stderr

        err := cmd.Run()
        if err != nil {
                return "", fmt.Errorf("command failed: %s
%s", err, stderr.String())
        }
        return strings.TrimSpace(stdout.String()), nil
}
```

4. LVM `internal/lvm/lvm.go`

```
package lvm

import (
        "fmt"
        "strconv"
        "strings"
        "syscall"
        "lvm-manager/pkg/utils"
)

// GetUsagePercent
func GetUsagePercent(mountPoint string) (int, error) {
        var stat syscall.Statfs_t
        err := syscall.Statfs(mountPoint, &stat)
        if err != nil {
                return 0, fmt.Errorf("failed to get fs stats for %s: %w", mountPoint, err)
        }

        total := stat.Blocks * uint64(stat.Bsize)
        free := stat.Bfree * uint64(stat.Bsize)
        used := total - free

        return int(float64(used) / float64(total) * 100), nil
}

// ExtendLV
func ExtendLV(lvPath string, incrementGB int) error {
        _, err := utils.RunCommand("lvextend", "-L", fmt.Sprintf("+%dG", incrementGB), lvPath)
        return err
}
```

```
// ResizeFS
func ResizeFS(lvPath string) error {
    //  (ext4/xfs)
    //  ext4
        _, err := utils.RunCommand("resize2fs", lvPath)
        return err
}
```

5. `internal/monitor/monitor.go`

```
package monitor

import (
        "fmt"
        "log"
        "time"
        "lvm-manager/internal/lvm"
)

type Target struct {
        LVPath           string `yaml:"lv_path"`
        MountPoint       string `yaml:"mount_point"`
        ThresholdPercent int    `yaml:"threshold_percent"`
        IncrementGB      int    `yaml:"increment_gb"`
}

func Start(targets []Target, interval time.Duration) {
        log.Println("Starting LVM auto-scaler...")
        ticker := time.NewTicker(interval)
        defer ticker.Stop()

        for {
                select {
                case <-ticker.C:
                        for _, target := range targets {
                                checkAndScale(target)
                        }
                }
        }
}

func checkAndScale(t Target) {
        usage, err := lvm.GetUsagePercent(t.MountPoint)
        if err != nil {
                log.Printf("ERROR: Failed to get usage for %s: %v", t.MountPoint, err)
                return
        }

        log.Printf("INFO: Usage for %s is %d%%", t.MountPoint, usage)

        if usage > t.ThresholdPercent {
                log.Printf("WARN: Usage %d%% > %d%%. Scaling up %s by %dGB.", usage, t.ThresholdPercent, t.LVPath, t.IncrementGB)

                if err := lvm.ExtendLV(t.LVPath, t.IncrementGB); err != nil {
                        log.Printf("ERROR: Failed to extend LV %s: %v", t.LVPath, err)
                        return
                }
                log.Printf("INFO: LV %s extended successfully.", t.LVPath)

                if err := lvm.ResizeFS(t.LVPath); err != nil {
                        log.Printf("ERROR: Failed to resize filesystem for %s: %v", t.LVPath, err)
                        return
                }
                log.Printf("INFO: Filesystem for %s resized successfully.", t.LVPath)
        }
}
```

1.

- **`resize2fs: Bad magic number in super-block`**: ext4
- **`lvextend: Insufficient free space`**: VG PE vgextend VG PV
- **`df -h`** : (resize2fs xfs_growfs)

2.

- : Go vgdisplay VG
- : JSON
- : Webhook

: `lvm-autoscaler`

1. : ResizeFS ext4 xfs
   - : `blkid -o value -s TYPE /dev/path`
2. **Dry-Run** : `--dry-run`
3. : `internal/lvm` Mocking
4. : LVM HTML Markdown VG/LV

1. :
    - : VG `vgextend` `lvm-autoscaler`
    - : LVM (`vgcfgbackup`) LV (`vgcfgrestore`)
2. :
    - **LVM** :
    - **LVM** : Day 3
3. : `lvm-manager` Go