Anleitung Throughput Measurements über GitHub CLI

throughput.ps1

1. Skript per powershell ausführen und repo angeben
2. Sollte Skript nicht funktionieren folgenden Code ausführen:

```
param([Parameter(Mandatory=$true)][string]$Repo, [int]$Days=60)

function WK([datetime]$d){
  $wday=[int]$d.DayOfWeek; if($wday -eq 0){$wday=7}
  $thr=$d.AddDays(4-$wday)

$week=[System.Globalization.CultureInfo]::InvariantCulture.Calendar.GetWeekOfYear($thr,[Syst
em.Globalization.CalendarWeekRule]::FirstFourDayWeek,[DayOfWeek]::Monday)
  "{0}-W{1:00}" -f $thr.Year,$week
}
function Skip($u){ if(!$u){return $false}; return ($u -match '\[bot\]$') -or ($u -eq
'github-classroom[bot]') }
$since=(Get-Date).ToUniversalTime().AddDays(-$Days)
$w=@{}
function Add($wk,$u,$f){
  if(Skip $u){return}; if(!$u){$u="(unknown)"}; $k="$wk|$u"
  if(!$w[$k]){$w[$k]=@{IssuesClosed=0;PRs=0;Commits=0}}
  $w[$k][$f]++
}

# IssuesClosed (Closer via Events, Fallback closed_by), nur seit $since
$q="repo:$Repo is:issue is:closed closed:>=$($since.ToString('yyyy-MM-dd'))"
$closed=gh api "search/issues?q=$([uri]::EscapeDataString($q))&per_page=100" --paginate --jq
'.items[] | {n: .number, t: .closed_at}' | ConvertFrom-Json
foreach($i in $closed){
  $wk=WK([datetime]$i.t)
  $ev=gh api "repos/$Repo/issues/$($i.n)/events?per_page=100" --paginate --jq '.[] |
select(.event=="closed") | {u:.actor.login, a:.created_at}' 2>$null | ConvertFrom-Json
  $closer= if($ev){ ($ev | Sort-Object {[datetime]$_.a} | Select-Object -Last 1).u } else {
$null }
  if(-not $closer){ $closer=gh api "repos/$Repo/issues/$($i.n)" --jq '.closed_by.login'
2>$null }
  Add $wk $closer 'IssuesClosed'
}

# PRs (nach created_at)
$prs=gh api "repos/$Repo/pulls?state=all&per_page=100" --paginate --jq '.[] |
{u:.user.login, t:.created_at}' | ConvertFrom-Json
foreach($p in $prs){ $dt=[datetime]$p.t; if($dt -ge $since){ Add (WK $dt) $p.u 'PRs' } }

# Commits (Autor oder Committer, nach Commit-Datum)
$cm=gh api "repos/$Repo/commits?since=$($since.ToString('o'))&per_page=100" --paginate `
  --jq '.[] | {u:(if .author then .author.login elif .committer then .committer.login else
null end), t:.commit.author.date}' | ConvertFrom-Json
foreach($c in $cm){ Add (WK ([datetime]$c.t)) $c.u 'Commits' }

# Ausgabe: eine Tabelle
$w.GetEnumerator() | Sort-Object {$_.Key} | %{
  $p=$_.Key -split '\|',2
  [pscustomobject]@{ Week=$p[0]; User=$p[1]; IssuesClosed=$_.Value.IssuesClosed;
PRs=$_.Value.PRs; Commits=$_.Value.Commits }
} | Sort-Object Week,User | Format-Table -AutoSize
```